The Next Phase of Eclipse's Development

Session 642

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

The Next Phase of Eclipse's Development

Daniel H Steinberg
Online Editor, O'Reilly Media
Bjorn Freeman-Benson
Technical Director OS Process and Infrastructure, Eclipse Foundation

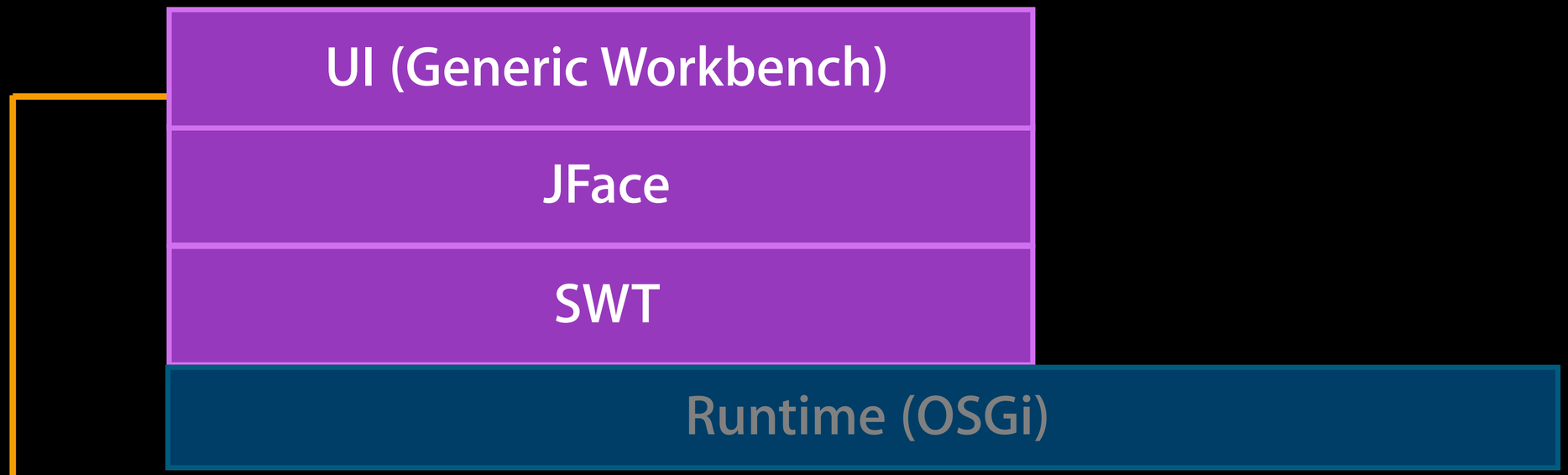
http://developer.apple.com/wwdc2005/

"Eclipse is an open platform for tool integration built by an open community of tool providers."

What Is Eclipse?

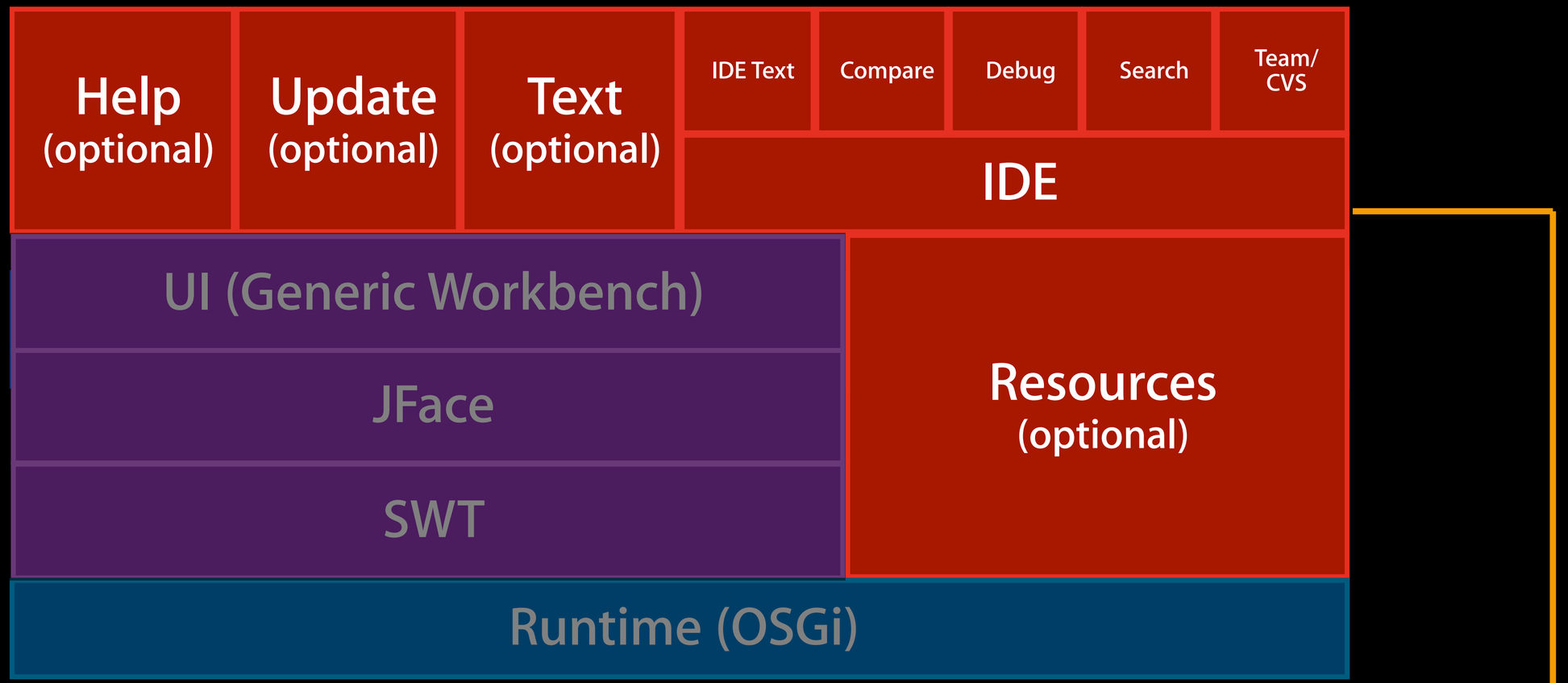
A client runtime

What Is Eclipse?



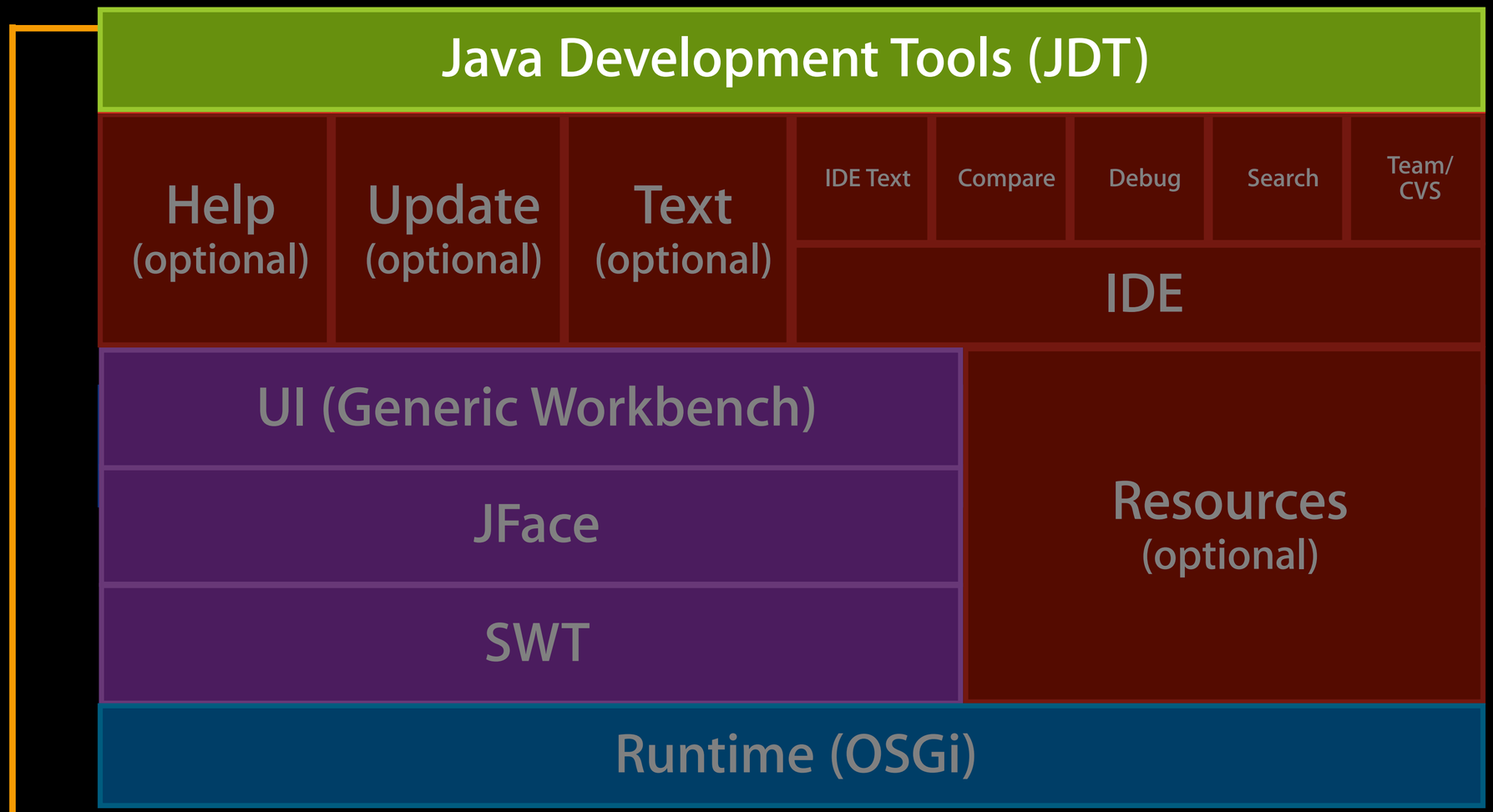
- An extensible application platform

What Is Eclipse?



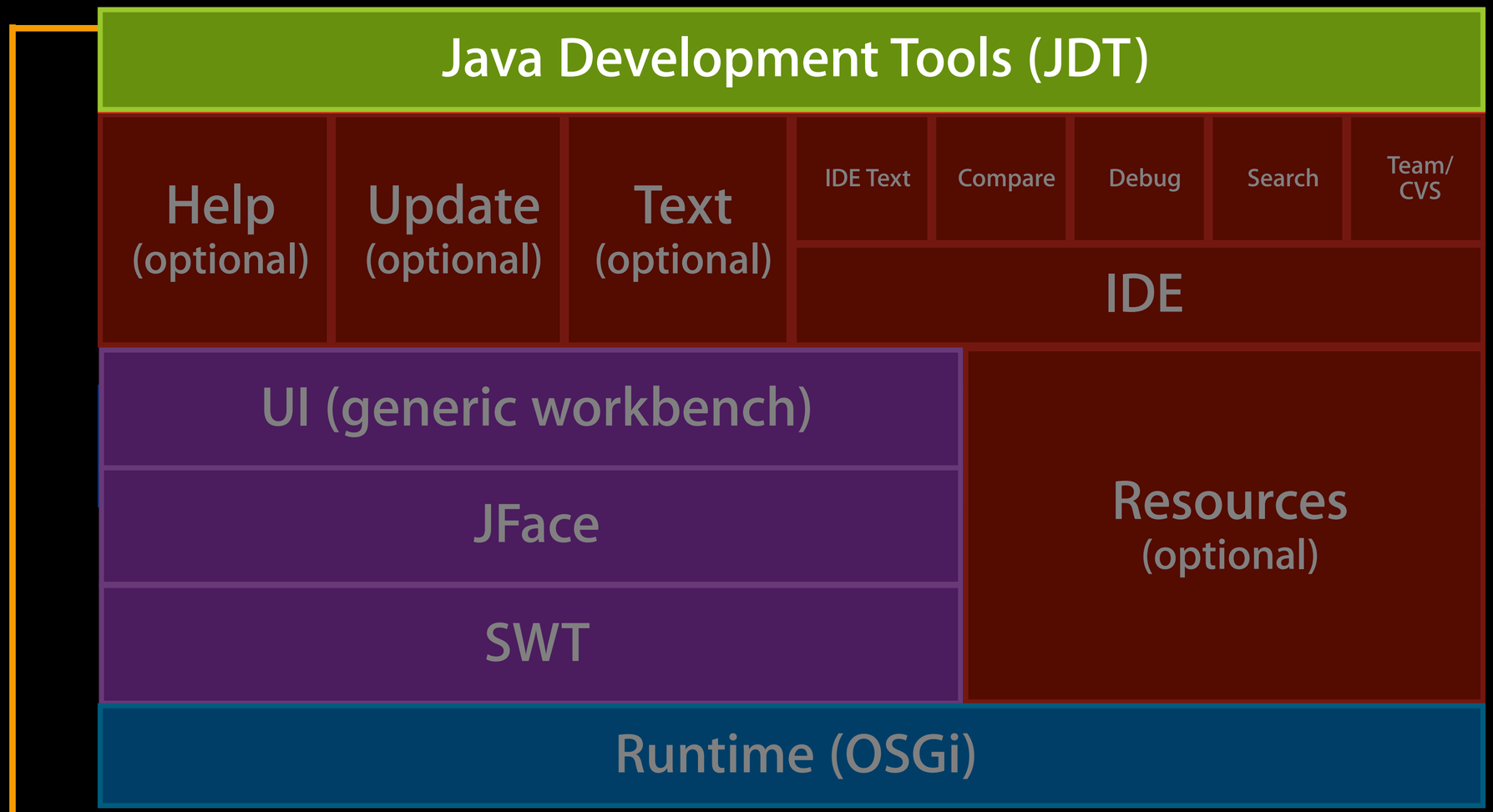
An extensible tool platform

What Is Eclipse?



A world-class Java IDE

What Is Eclipse?

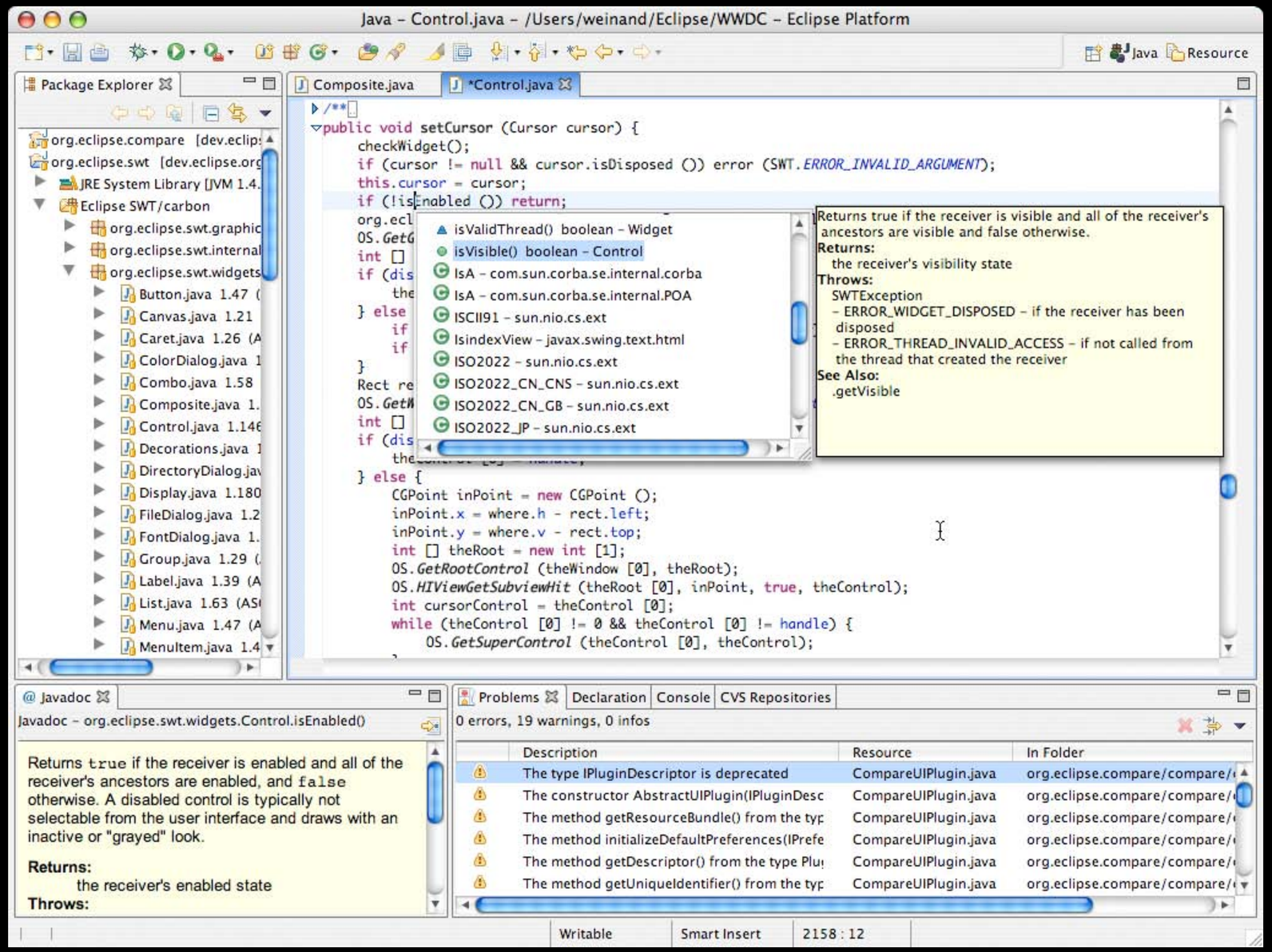


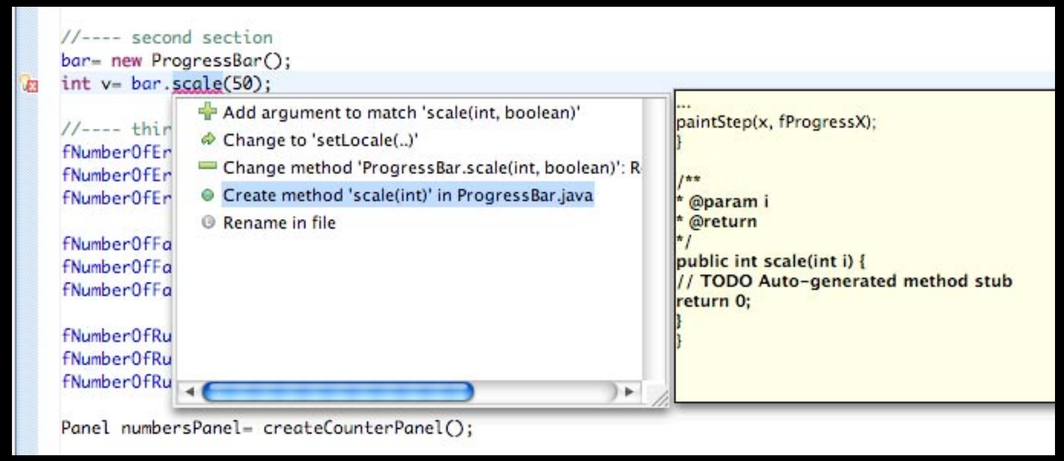
- Today's presentation will focus on the JDT

What Is JDT?

http://developer.apple.com/wwdc2005/

Java Development Tools (JDT)

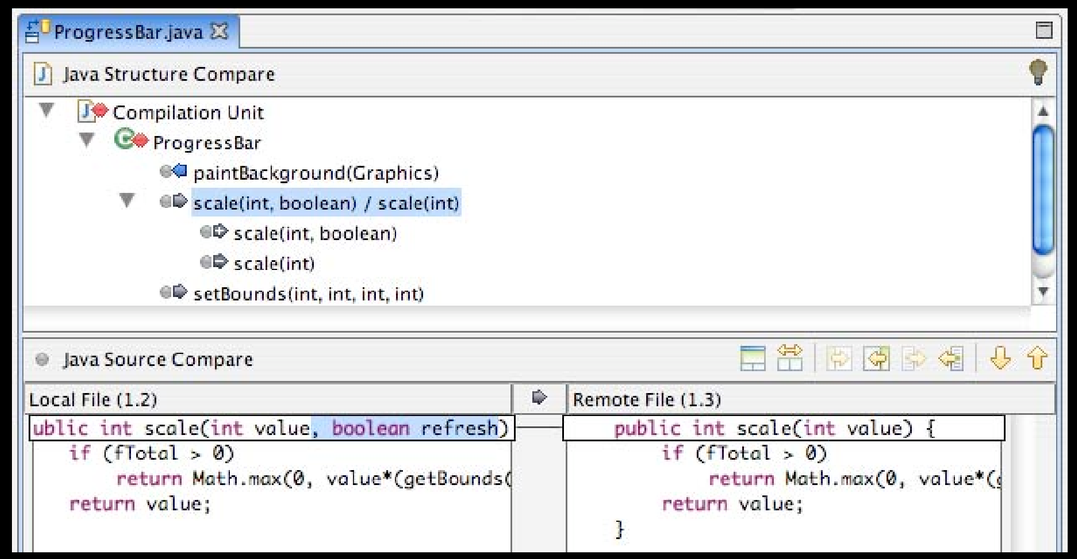




JDT Features

- Incremental automatic building in the background
- Debugging
  - Hot code replace
  - Expression evaluation
  - Run code with errors
- Refactoring
- Coding assistance
  - Code completion
  - Quick fix/quick assist
  - Import assist
  - Templates
  - Linked editing

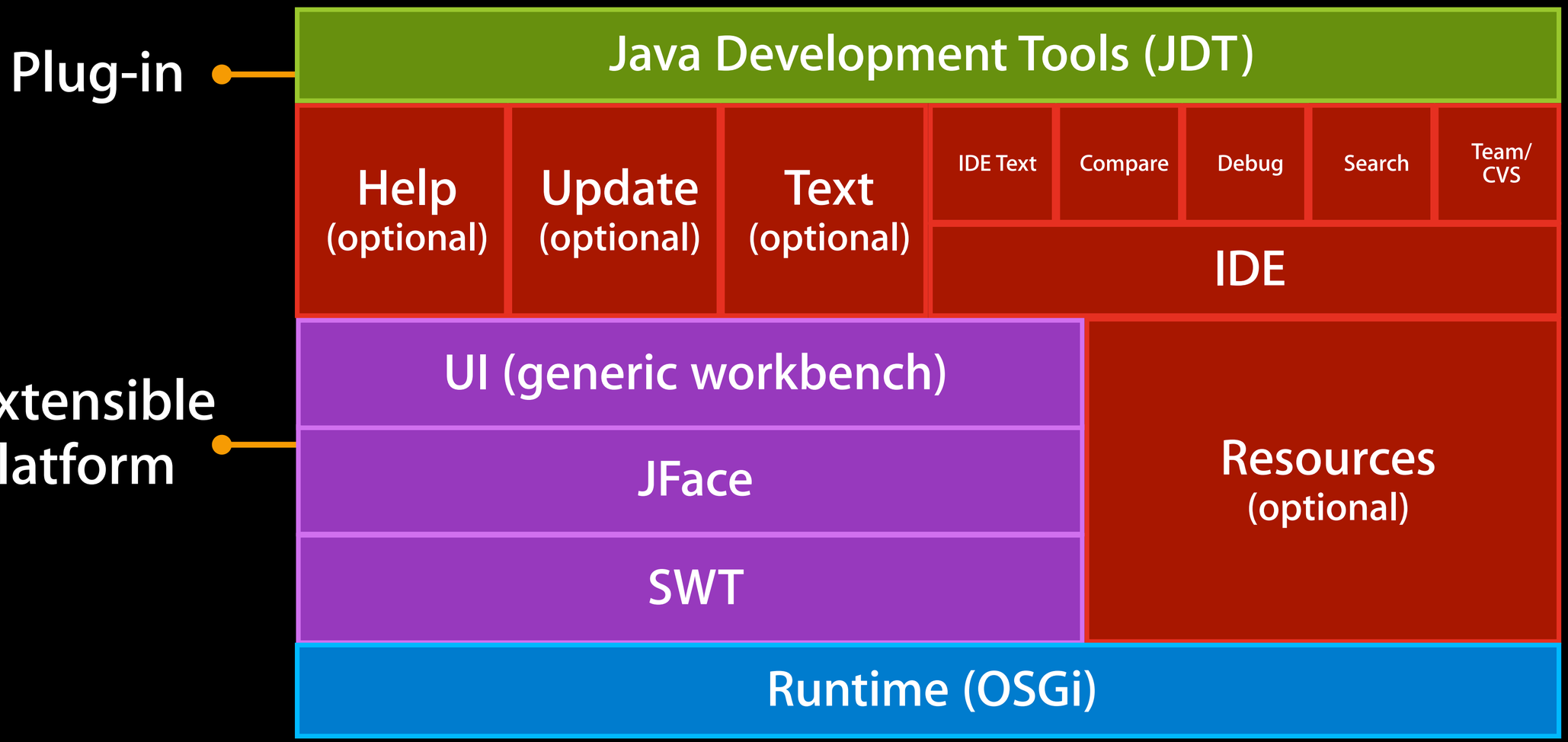
JDT Features (Cont.)



- Precise searching
  - References
  - Type hierarchies
  - All types
- Java-aware structural comparisons/local history
- Open source tools integration
  - Ant
  - JUnit
- Extension APIs

JDT Is a Set of Eclipse Plug-Ins

Extends Core Platform Functionality



What’s New in 3.1?

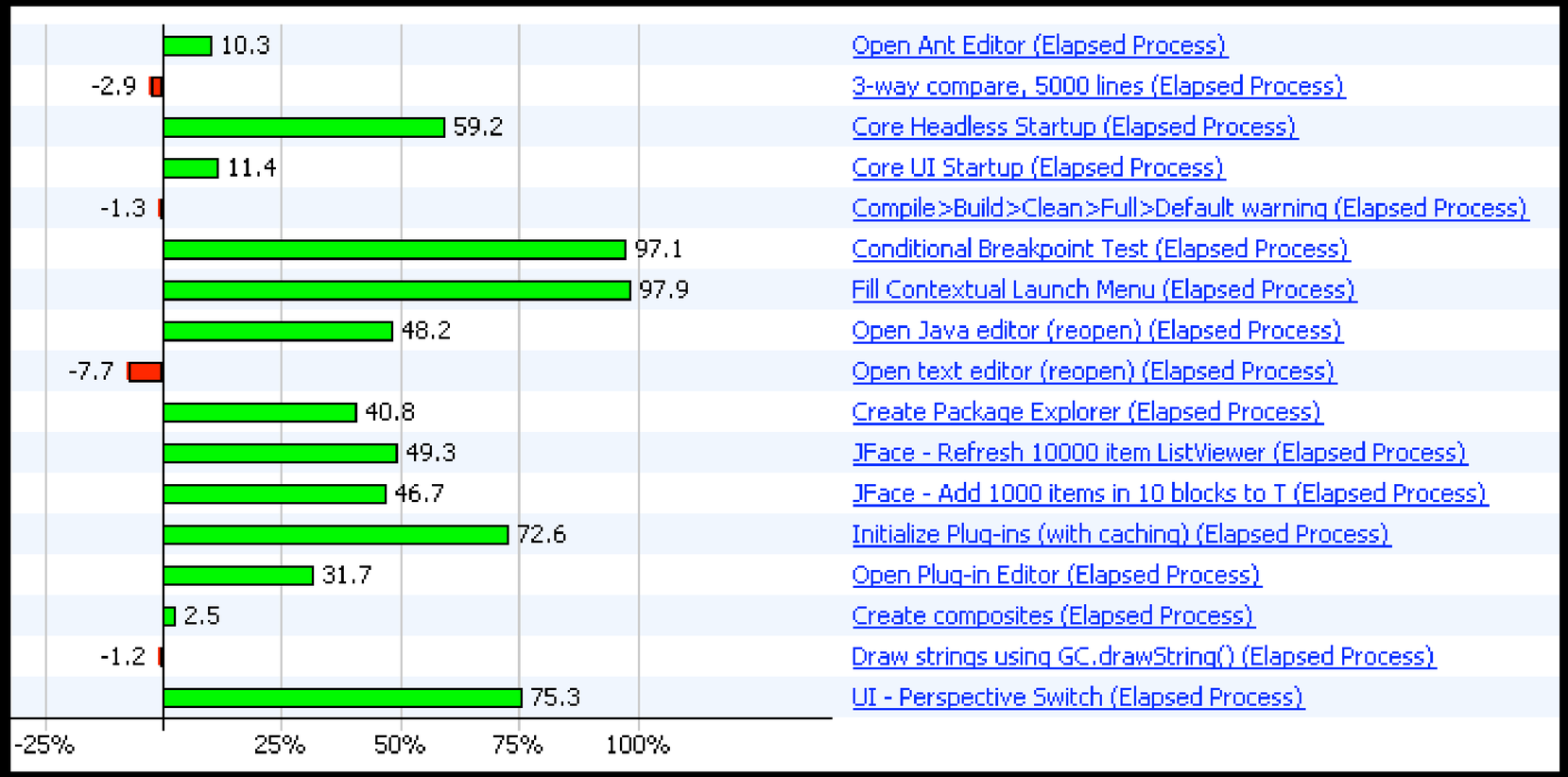
http://developer.apple.com/wwdc2005/

Themes for 3.1

- Scaling up
- Simple to use
- 3.1 runtime
- J2SE 5 support
- Enterprise ready
- Design for extensibility: be a better platform

Scaling Up

- Large-scale work spaces
- Performance…
  - Continuous performance monitoring
  - Instrumentation



Simple to Use

- Preferences
- User assistance—dynamic help
- JDT—lots of new quick assists
- Enhancements requests with lots of votes

What's New in the 3.1 Runtime

- Enhanced OSGi support
  - Bundle permissions
- Java Web Start (JNLP) support
- Plug-ins as jars
- Security
  - Jar signing

J2SE 5.0

- “deep” support in refactoring, search, and quick fix
- Autoboxing highlighting
- Generics in refactoring
- Quick assist to convert to enhanced for loop

Enterprise Ready

- Working in large workspaces
- Introduced working sets in the package explorer
- Shared settings per project
- Import/export of settings
- Architectural access restrictions
- nls—hovering and linking
- JUnit—large-scale test suites and test prioritization
- ant—import/export and debug

Design for Extensibility—Be a Better Platform

- Refactoring of action contributions
- New APIs
  - Generalized undo/redo
  - Content types
  - ltk (language toolkit) enhancements
- Push down JDT-specific functionality
  - Editor linking

Eclipse 3.1

- Open source development environment
- Go to www.eclipse.org
- Built in Java with Carbon library (SWT)
- Zero code changes to build universal
  - SWT built without modification
  - Minor makefile changes

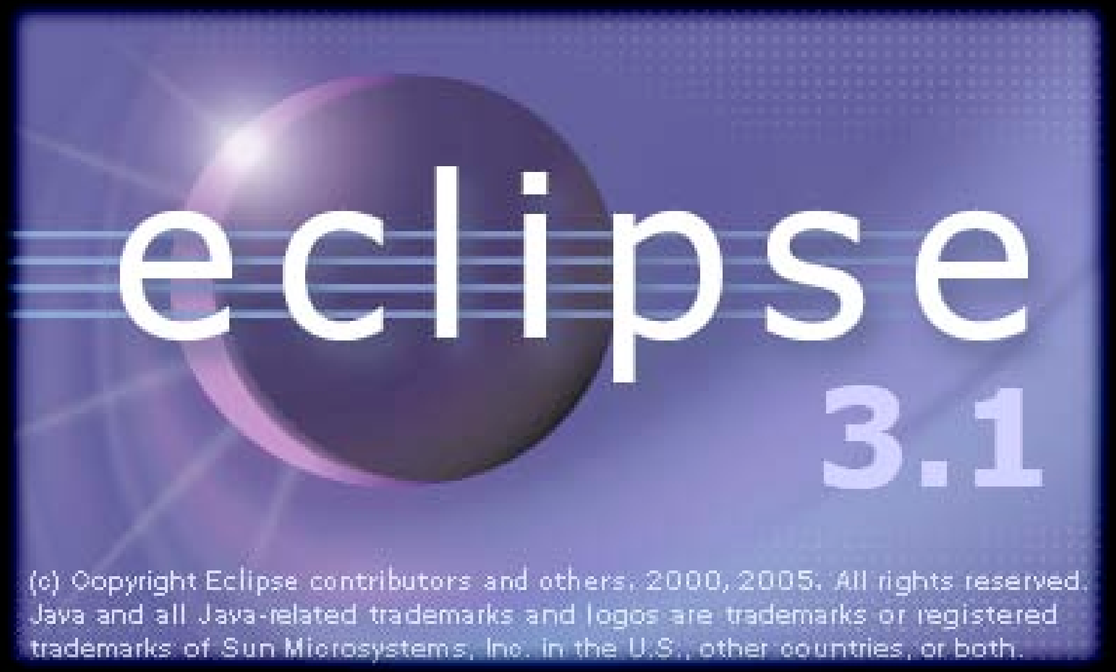
What You'll Learn This Afternoon

- Getting started
- Packages and classes
- Programming by intention
- Code assist
- Running tests
- J2SE 5.0
- Building plug-ins

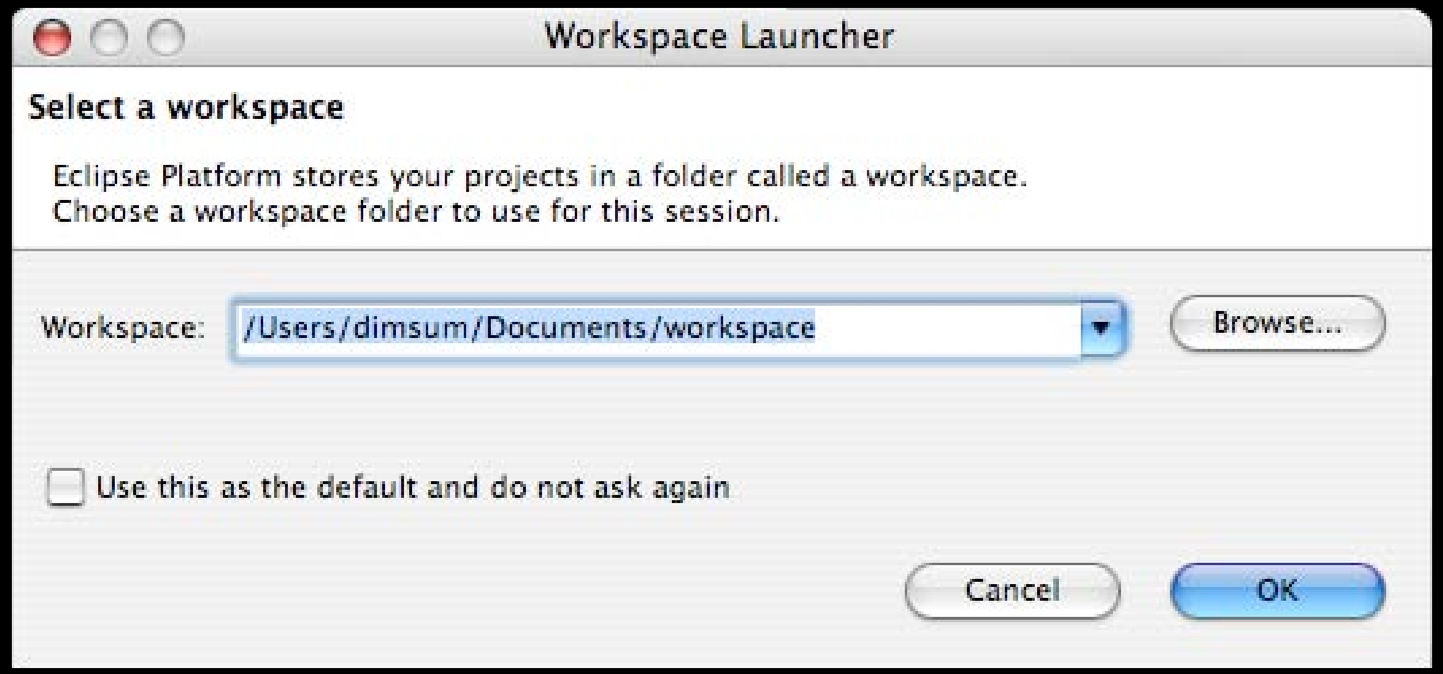
Getting Started

http://developer.apple.com/wwdc2005/

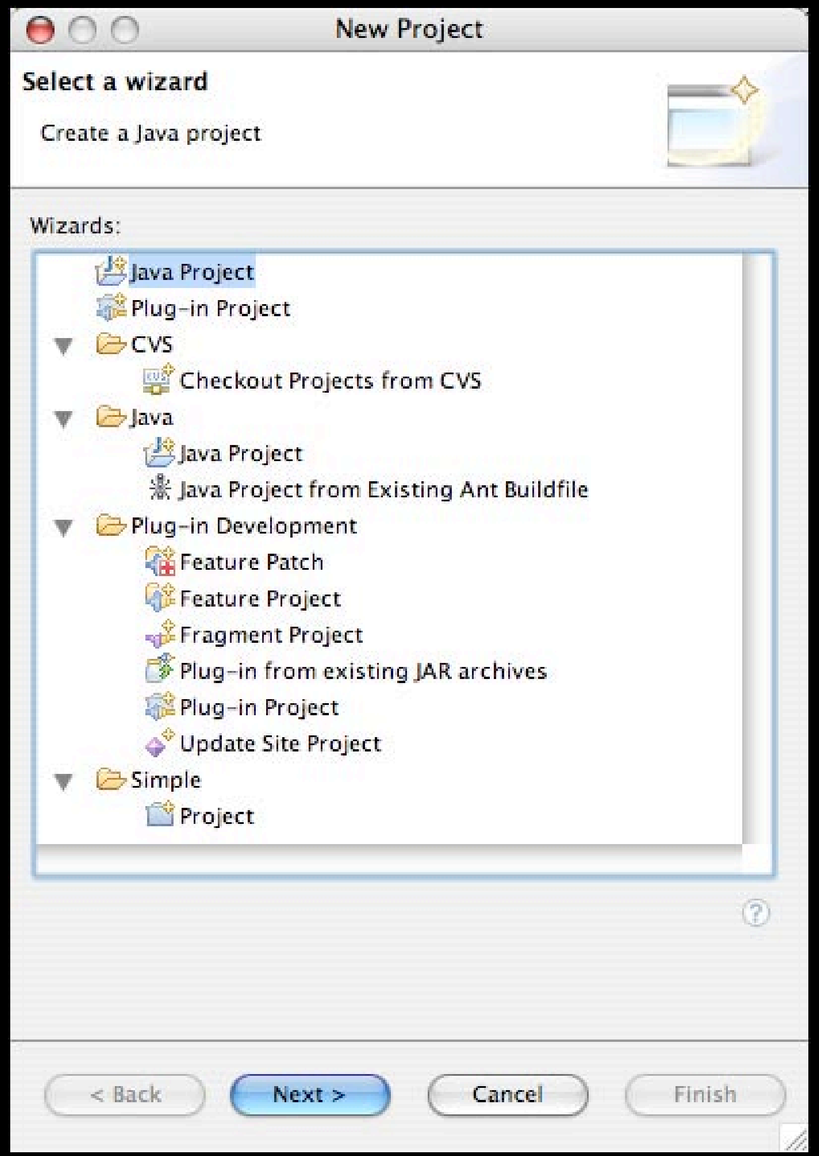
Using Eclipse



Choosing a Workspace



Creating a Project



Keyboard Shortcuts: Command-Shift-L

Activate Editor	F12
Add Block Comment	⇧⌘/
Add Import	⇧⌘M
Add Javadoc Comment	⌥⌘J
Backward History	⌥←
Breakpoints	⇧⌥Q B
Build All	⌘B
Change Method Signature	⌥⌘C
Cheat Sheets	⇧⌥Q H
Close	⌘W
Close All	⇧⌘W
Collapse	⌘Numpad_Subtract
Console	⇧⌥Q C
Content Assist	^␣
Context Information	⇧⌘␣
Convert Local Variable to Field	⌥⌘F
Copy	⌘Insert
Copy Lines	⌥⌘↓

Press "⇧⌘L" to open the preference page.

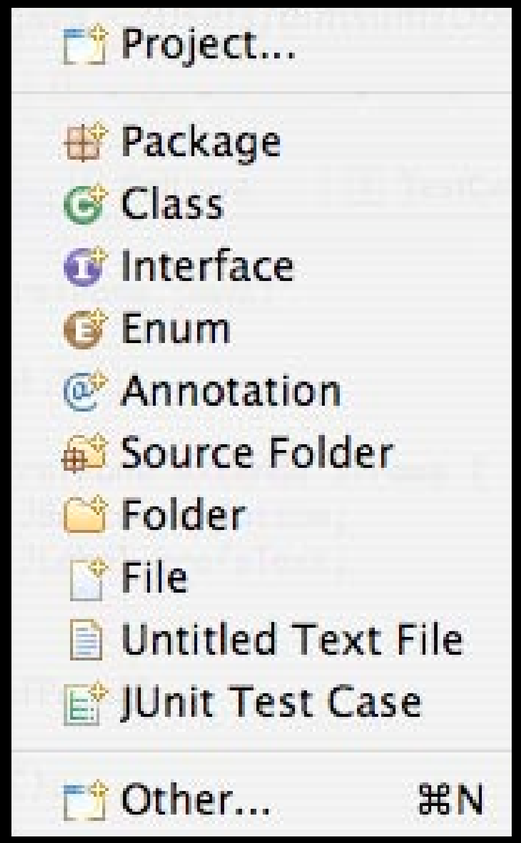
Getting Started Hands-on Demo

- Create swing project in Xcode
- Import it into Eclipse
- Look at preferences to turn off warning
- Turn on highlighting to explore types
- Note default package

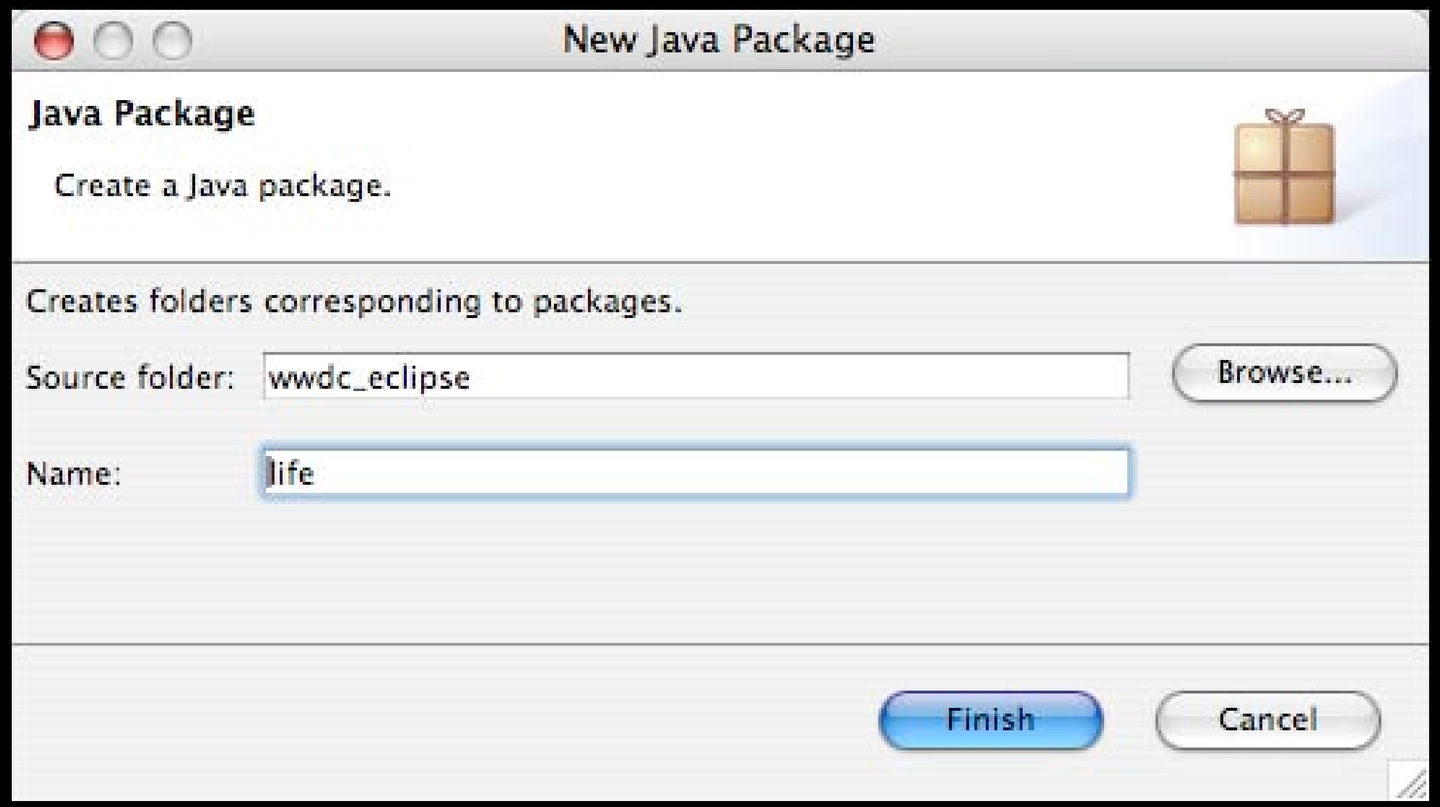
Packages and Classes

http://developer.apple.com/wwdc2005/

New Creations

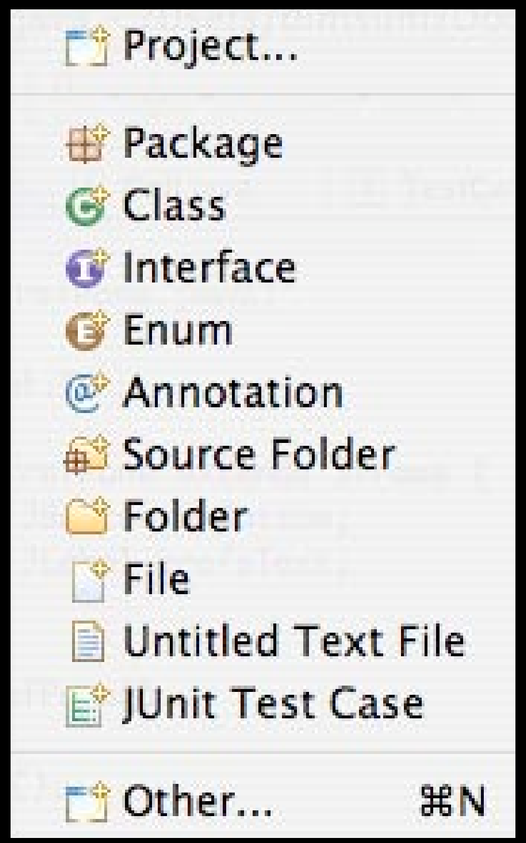


Package

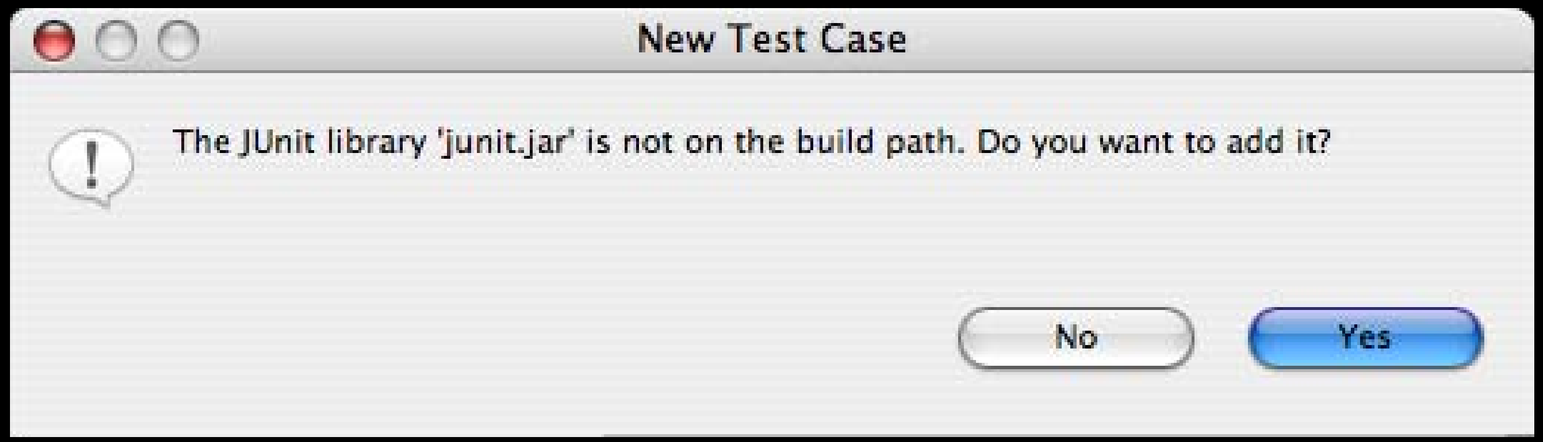


Eclipse understands and supports the correspondence between the directory structure and the type hierarchy

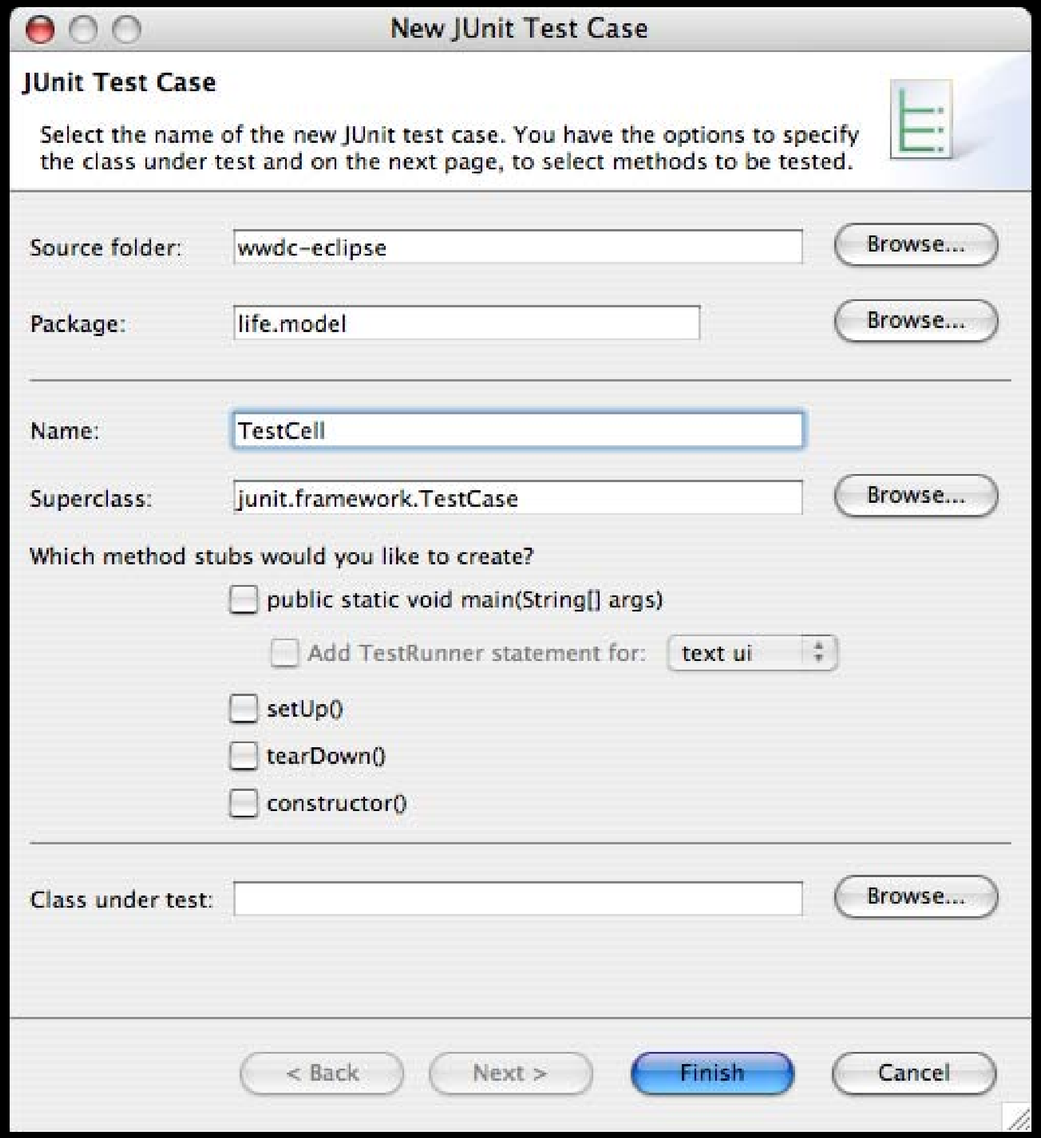
Creating a Test Case

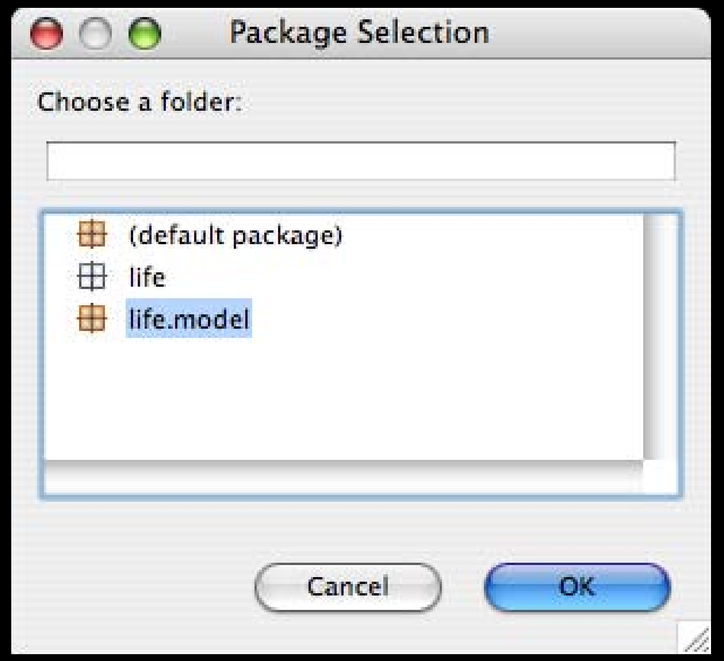


Helpful Hints



Configuring a Test





Generated Test Class

```
package life.model;

import junit.framework.TestCase;

public class TestCell extends TestCase {

}
```

Packages and Classes Hands-on Demo

- Move classes from default package
- Create new package
- Create Help menu
- Create Don’t Panic menu item

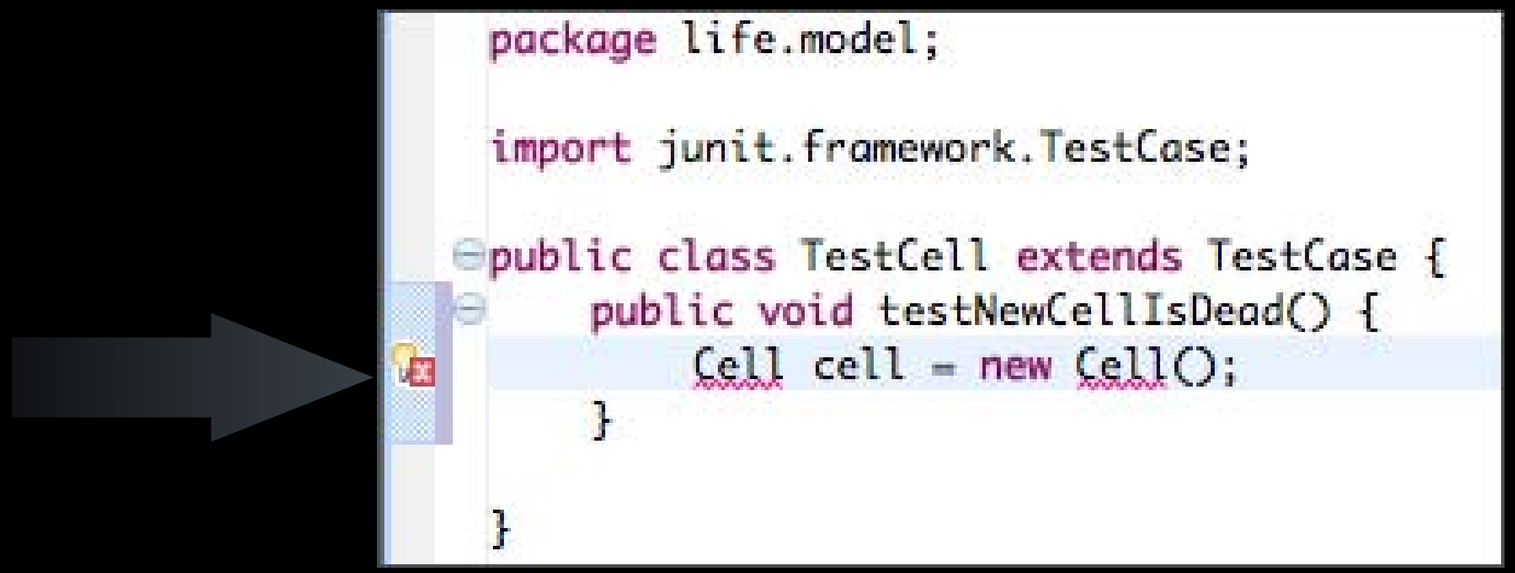
Programming by Intention

http://developer.apple.com/wwdc2005/

Context-Sensitive Help

- “Lightbulb”
- Source hints
- Refactorings

Pretend That What You Want to Exist Does Exist



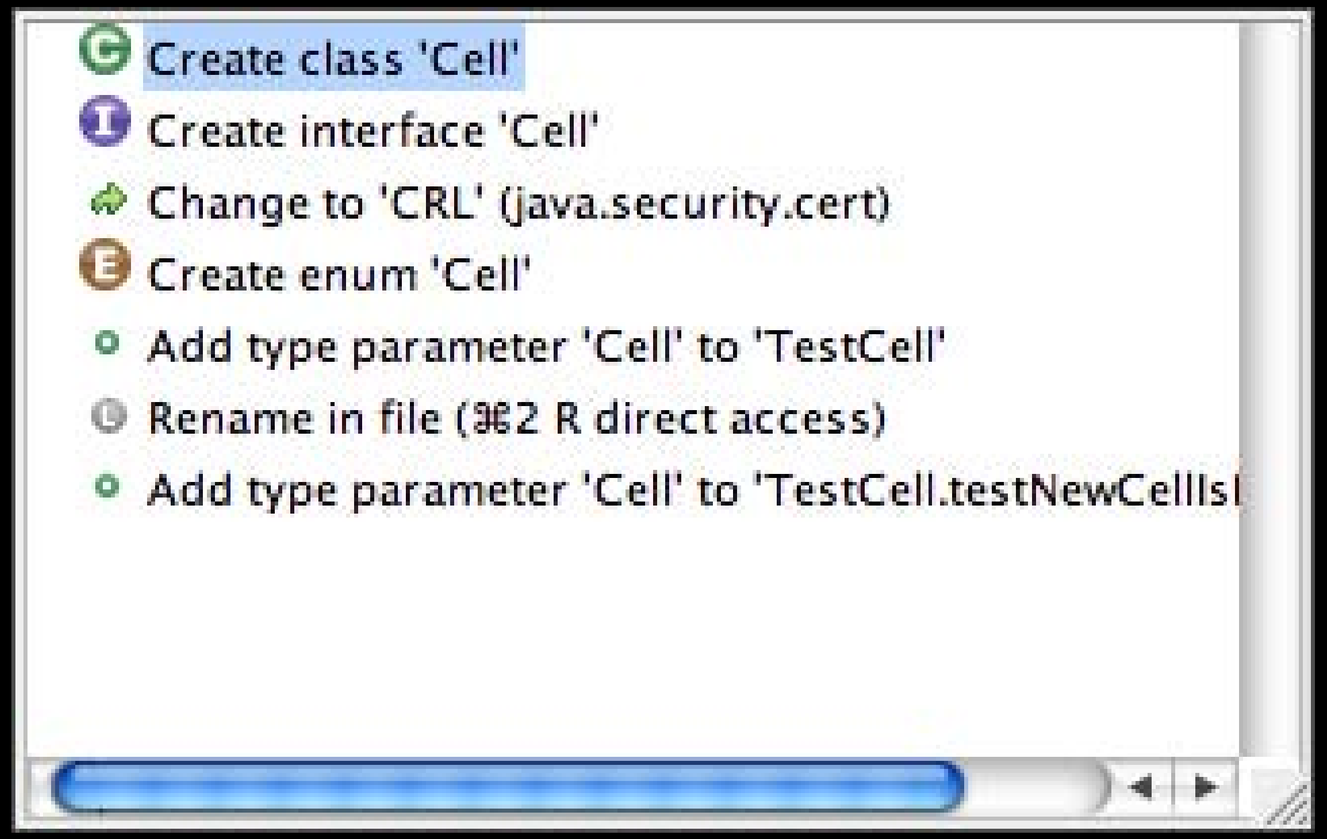
```
package life.model;

import junit.framework.TestCase;

public class TestCell extends TestCase {
    public void testNewCellIsDead() {
        Cell cell = new Cell();
    }

}
```

Squiggly lines and light bulb—
accept the help you are being offered

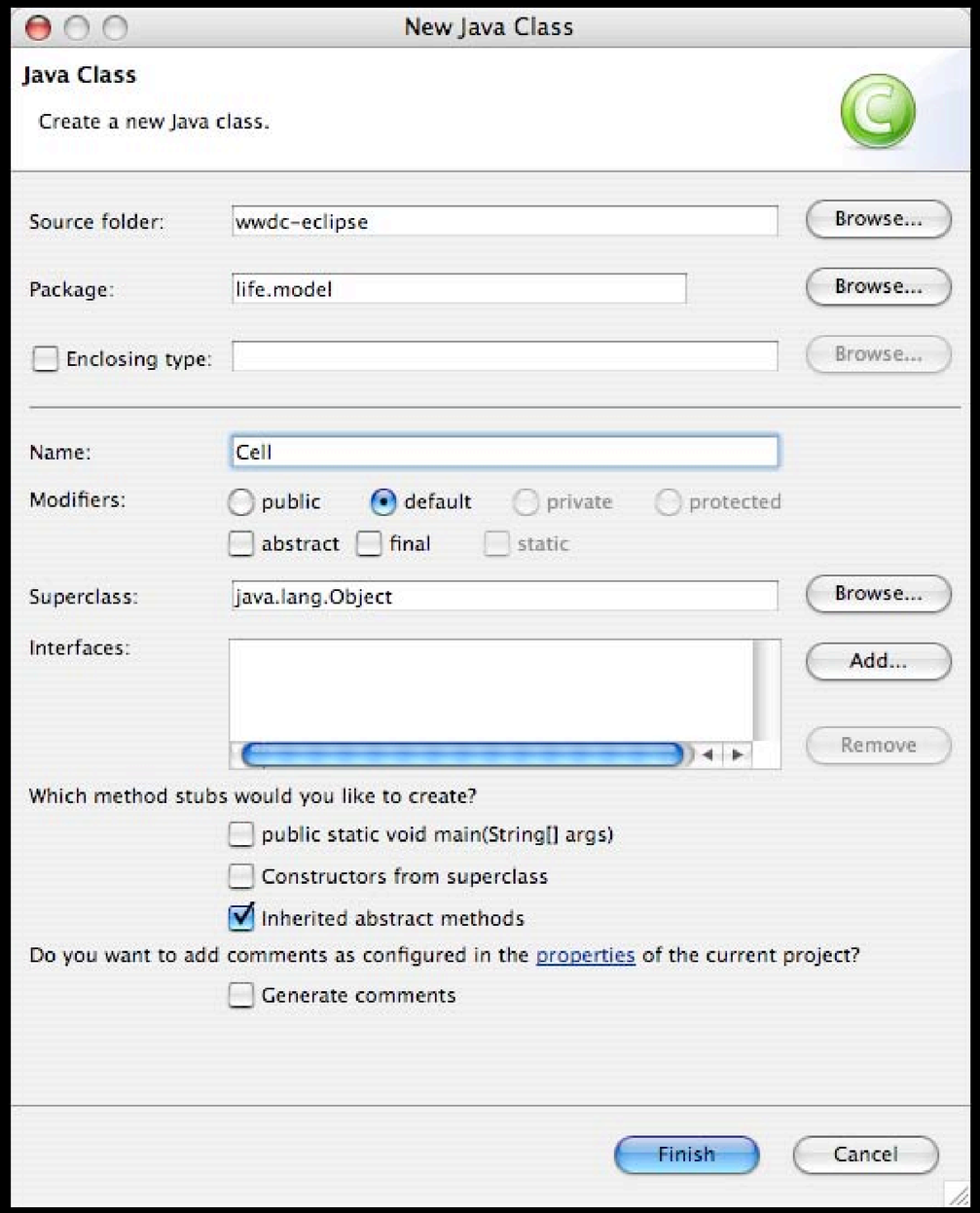
What's Wrong With Cell cell = new Cell()

- The Cell class doesn't exist
- The no argument constructor doesn't exist
  (although fixing the first would fix the second)
- The code won't compile

How Could We Fix It?



Choose to Create Cell



Generated Cell Class

```
package life.model;

class Cell {

}
```

Why is this better than a text editor?

Intention Hands-on Demo

- Return to the Help menu
- Use intention to add a new Find Towel menu item

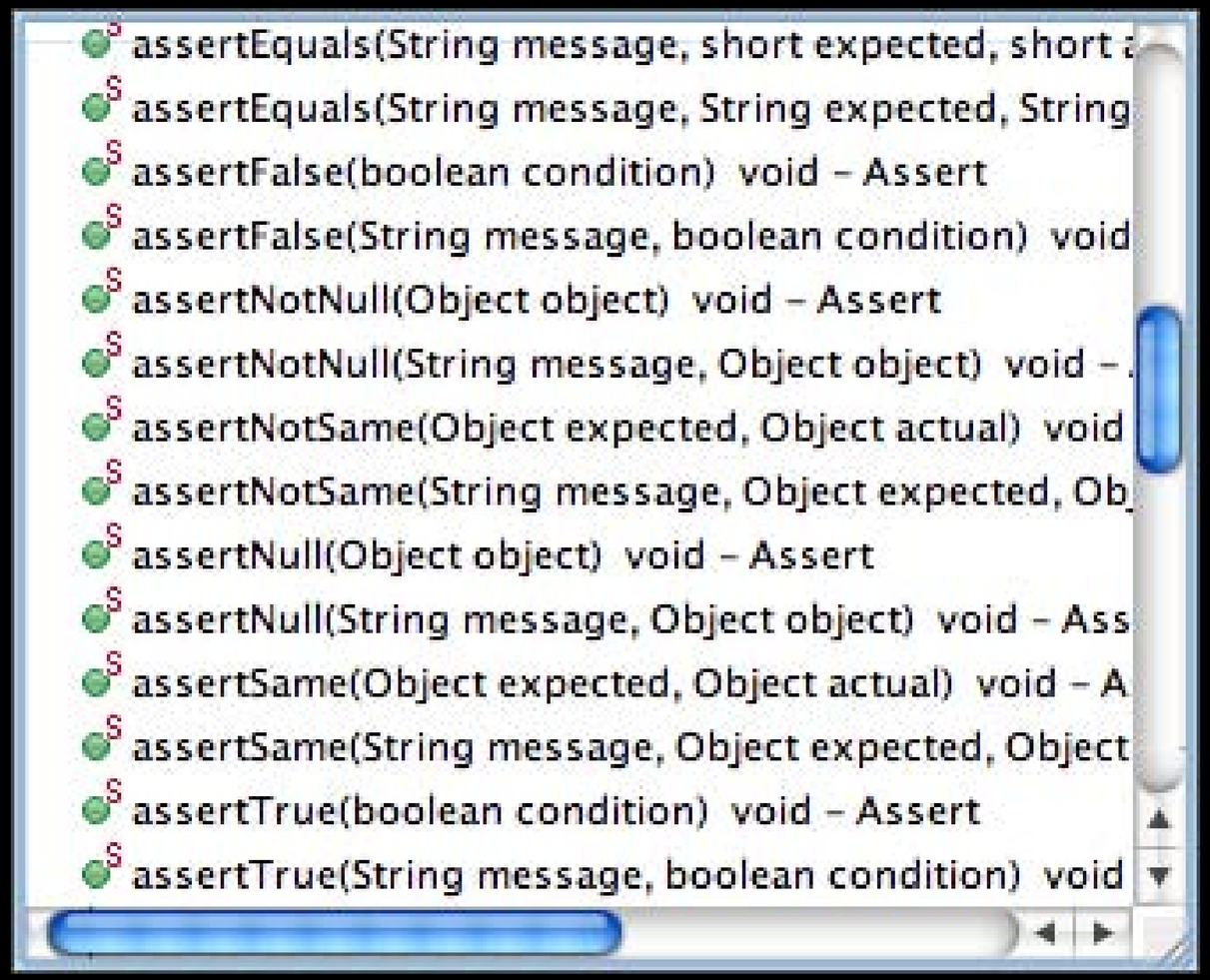
Code Assist

http://developer.apple.com/wwdc2005/

Activating

- When help is obviously needed:
  - `JFrame myFrame = new JFrame();`
  - `myFrame.`
- In response to CTRL- space

Code Assist in TestCell.testNewCellIsDead()



Pop-up Help for Code Completion

The Best Intentions

```
public void testNewCellIsDead() {
    Cell cell = new Cell();
    assertTrue("Cell should be dead when created.",
               !cell.isAlive());
}
```

Previewing the Changes

```
...
class Cell {

public boolean isAlive() {
// TODO Auto-generated method stub
return false;
}
}
```

- Create method 'isAlive()' in type 'Cell'
- Add cast to 'cell'
- Rename in file (⌘2 R direct access)

The Generated Method in the Cell Class

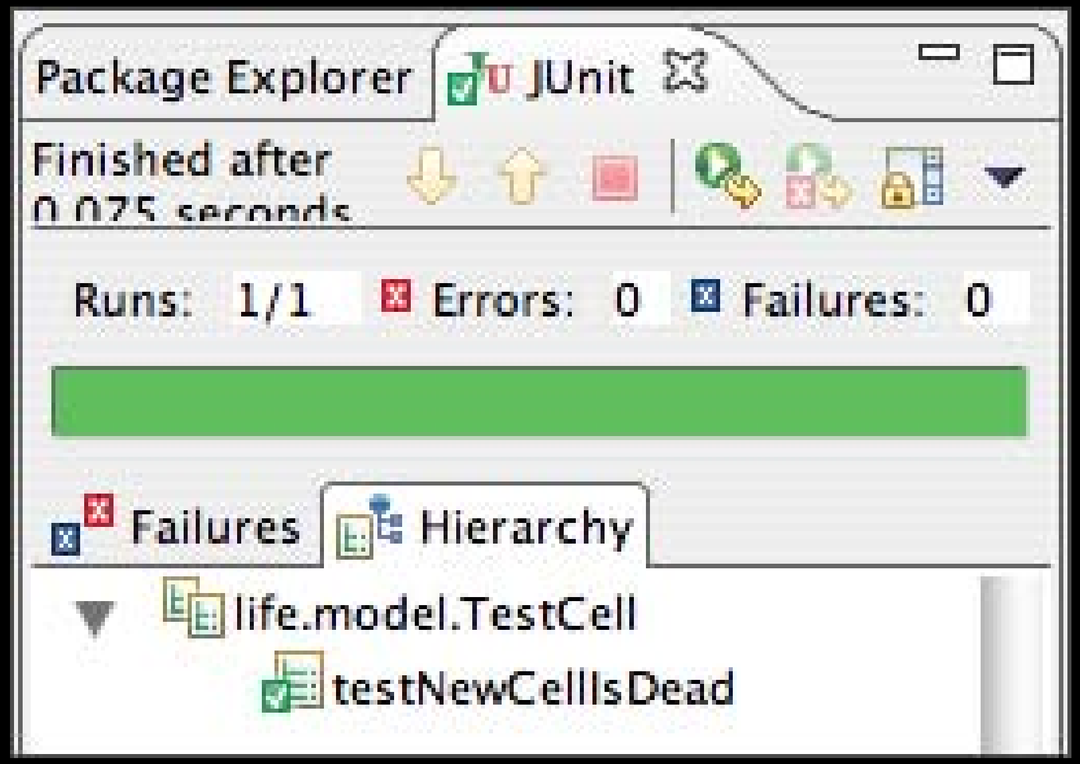
```
package life.model;

class Cell {

    public boolean isAlive() {
        // TODO Auto-generated method stub
        return false;
    }

}
```

Code Assist Hands-on Demo

- Run GUI code
- Locate the beginning of the constructor for AboutBox
- Insert a line after the call to `super("");`
- Use code assist to find `setBackground()` and set the background color to `Color.CYAN`
- Rerun GUI code

Running Tests

http://developer.apple.com/wwdc2005/


Package Explorer
JUnit
Finished after
0.075 seconds
Runs: 1/1
Errors: 0
Failures: 0
Failures
Hierarchy
life.model.TestCell
testNewCellIsDead

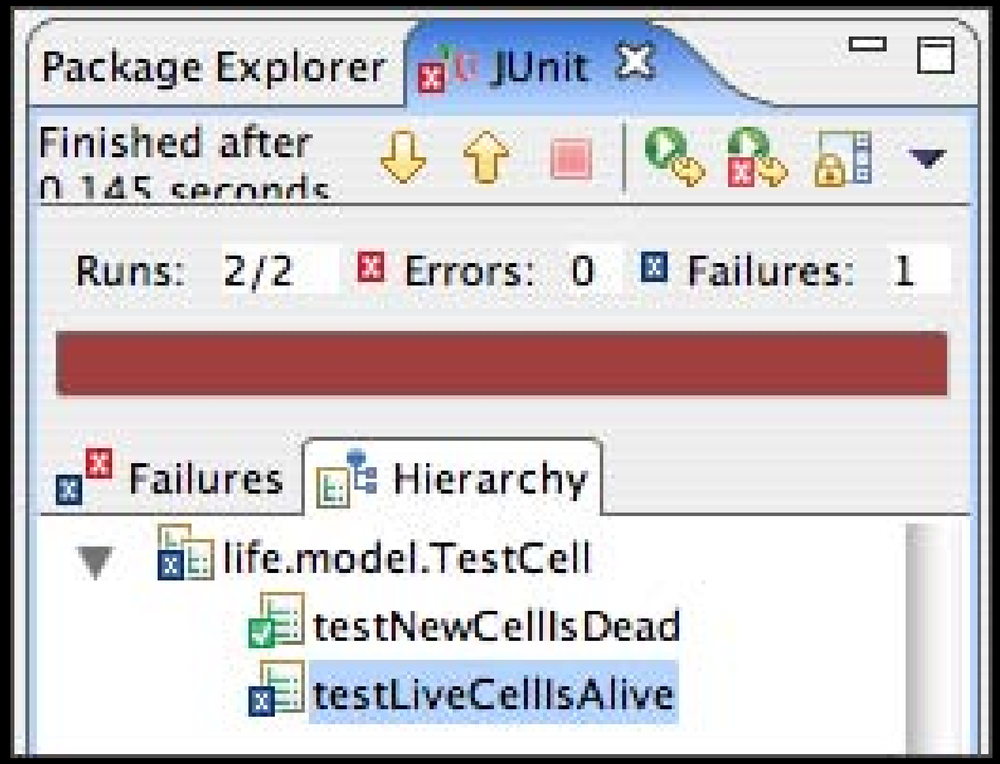
Is a Live Cell Alive?

```
    public void testLiveCellIsAlive(){
        Cell cell = new Cell();
        cell.setAlive(true);
        assertTrue(cell.isAlive());
    }

}
```

Run Test as Soon as Code Compiles

- Need to add "`setAlive()`" method to Cell:
  - `public void setAlive(boolean isAlive){}`
- Note—it doesn't need to do anything
- Recall the `isAlive()` method always returns `false`
  - This used to be a good thing

Test Cycle


Package Explorer
JUnit
Finished after
0.145 seconds
Runs: 2/2
Errors: 0
Failures: 1
Failures
Hierarchy
life.model.TestCell
testNewCellIsDead
testLiveCellIsAlive

Fixing Cell

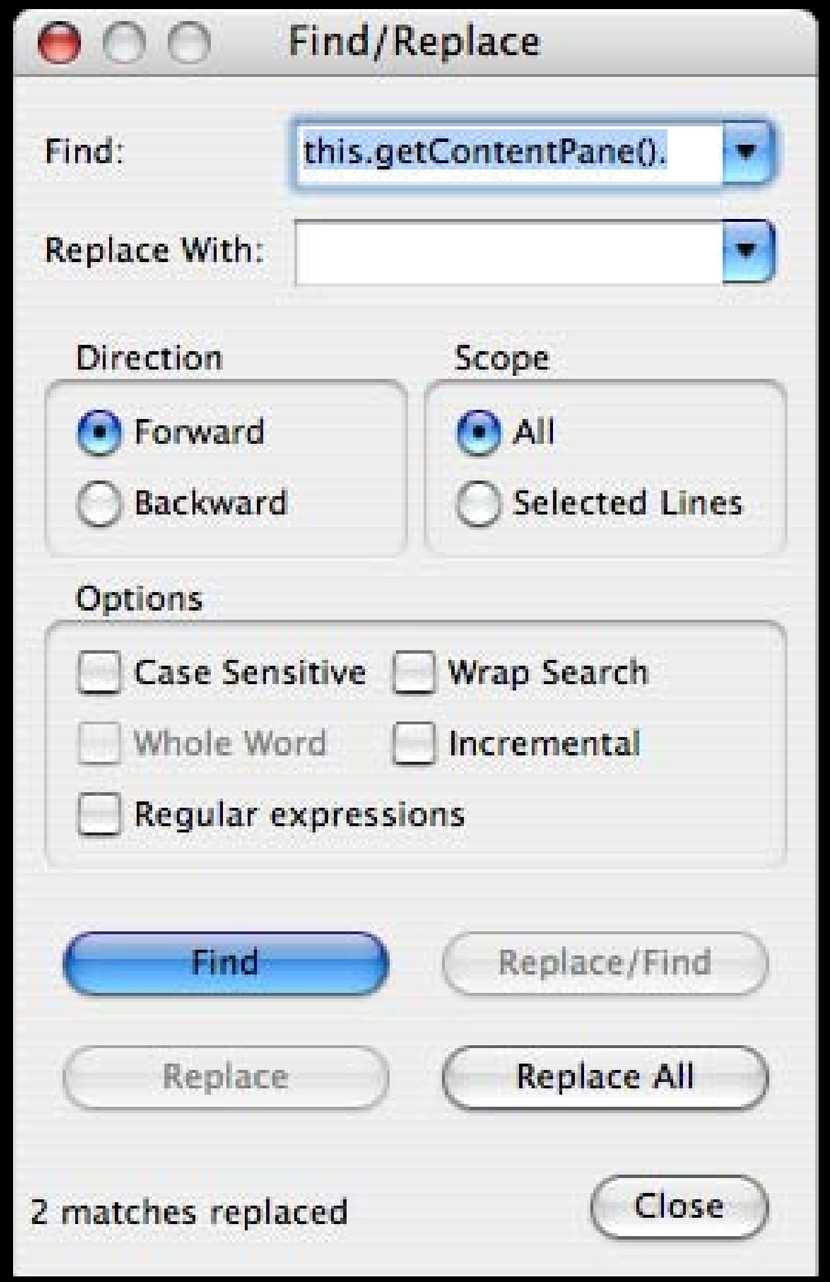
```
package life.model;

class Cell {

    private boolean alive;

    public boolean isAlive() {
        return alive;
    }

    public void setAlive(boolean alive) {
        this.alive = alive;
    }

}
```

Highlighting Is Smart

```
package life.model;

class Cell {

    private boolean alive;


    public boolean isAlive() {
        return alive;
    }

    public void setAlive(boolean alive) {
        this.alive = alive;
    }

}
```

Not all “alive”s are the same

Find, Replace, and Reg Ex



Repeated Code

```
public void testNewCellIsDead() {
    Cell cell = new Cell();
    assertTrue(!cell.isAlive());
}

public void testLiveCellIsAlive(){
    Cell cell = new Cell();
    cell.setAlive(true);
    assertTrue(cell.isAlive());
}
```

Make the local variable cell into a field
and extract the instantiation into a method

Resulting Code

```
package life.model;

import junit.framework.TestCase;

public class TestCell extends TestCase {

    private Cell cell;

    protected void setUp() {
        cell = new Cell();
    }

    public void testNewCellIsDead() {
        setUp();
        assertTrue(!cell.isAlive());
    }

    public void testLiveCellIsAlive() {
        cell.setAlive(true);
        assertTrue(cell.isAlive());
    }
}
```

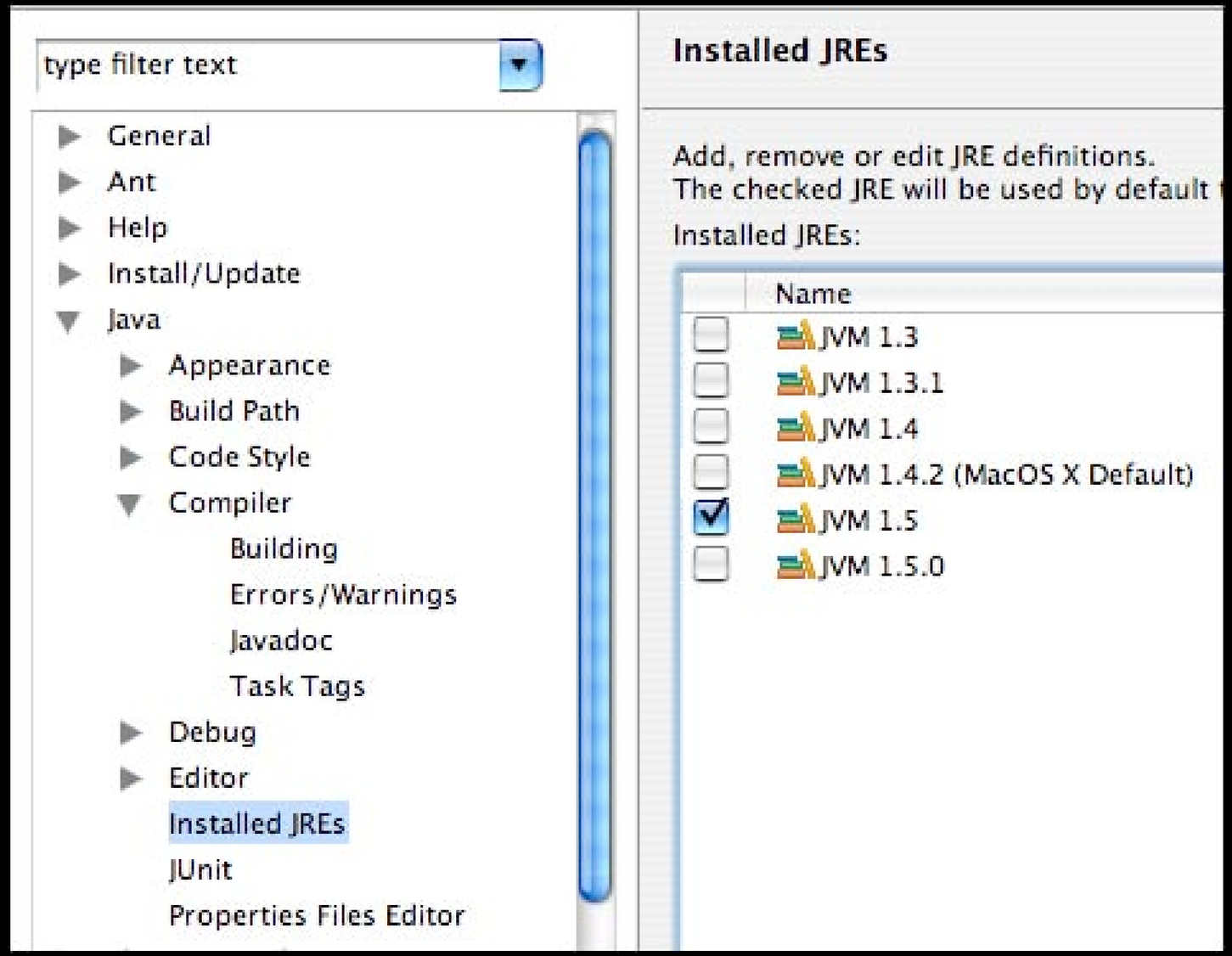
Testing Cycle Hands-on Demo

- Open the class PrefPane.java
- Ensure Eclipse preferences are set to Java 5.0
- Do a find and replace removing `this.getContentPane()`
- Rename the JLabel `prefsText` to `preferencesText`
- Add an `import` statement for `javax.media.jai.*;`

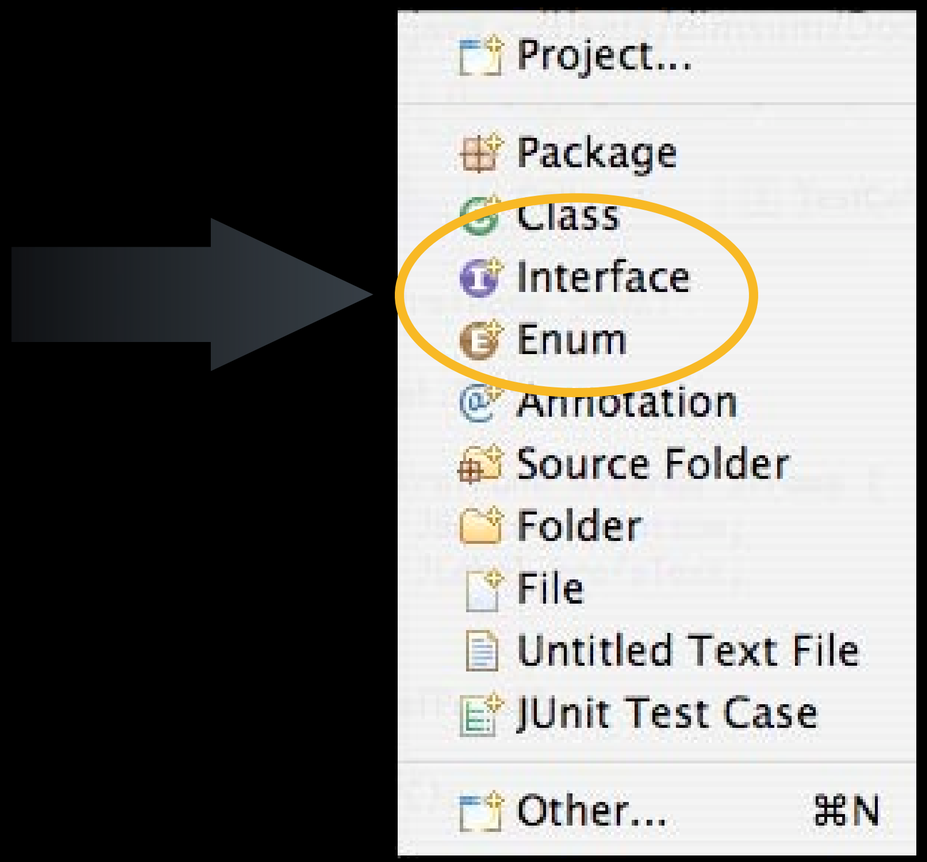
J2SE 5.0 With Eclipse

http://developer.apple.com/wwdc2005/

Setting VM Preferences



Enums and Annotations



Generics

```
package life.model;

import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;
```

Type safety: The method next() belongs to the raw type Iterator. References to generic type Iterator<E> should be parameterized

```
  private static List squares = new ArrayList();

   private static void fillList() {
    for (int i = 0; i < 6; i++) {
      squares.add(i * i);
    }
  }

  private static void outputList() {
    Iterator iterator = squares.iterator();
    int j=0;
    for (; iterator.hasNext();) {
      System.out.printf("%d squared is %d.\n",
                        j++, iterator.next());
    }
  }
}
```

Generics

Move... ⌥⌘V
Pull Up...
Push Down...
Extract Interface...
Use Supertype Where Possible...
Infer Generic Type Arguments...
Extract Local Variable... ⌥⌘L
Extract Constant...
Introduce Parameter...

Introduce Type

```
private static List<?> squares = new ArrayList<Object>();

 private static void fillList() {
  for (int i = 0; i < 6; i++) {
    squares.add(i * i);
  }
```

Manual Cleanup

```
private static List<Integer> squares = new ArrayList<Integer>();

 private static void fillList() {
  for (int i = 0; i < 6; i++) {
    squares.add(i * i);
  }
}
```

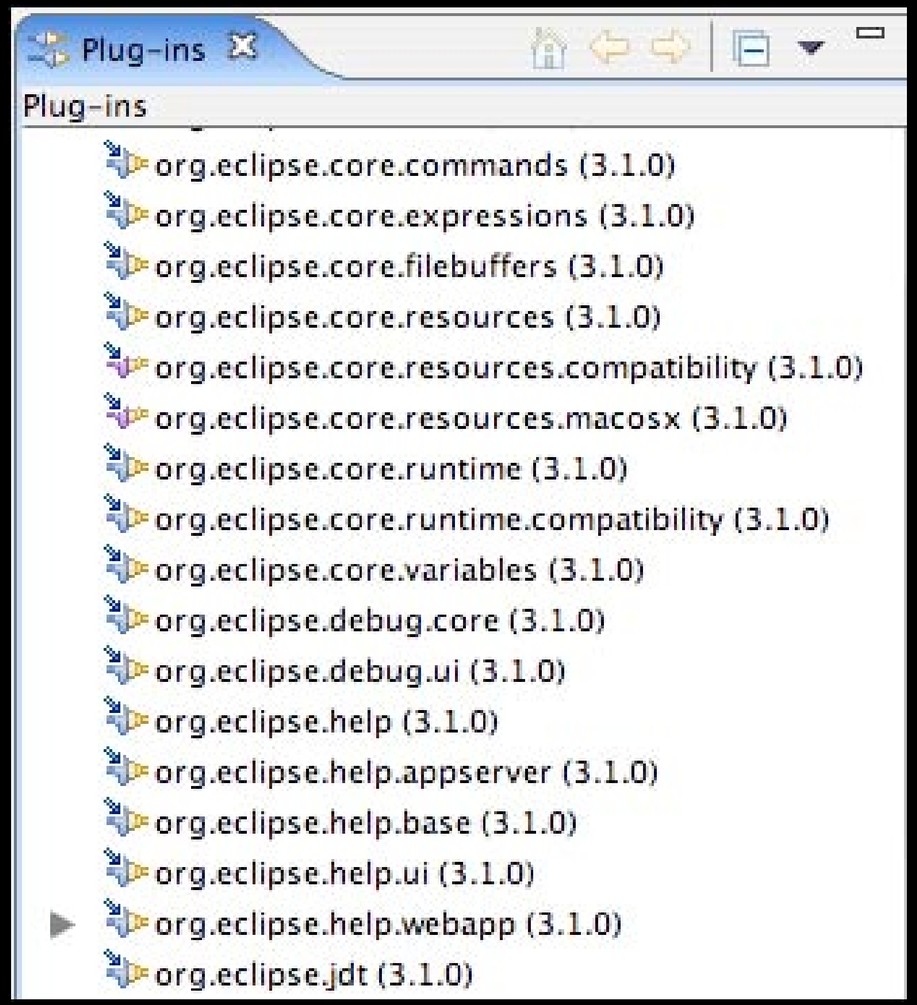
J2SE 5.0 Hands-on Demo

- Open AboutBox.java
- Locate the array of JLabels
- Introduce an ArrayList and use generics
- Add an enhanced for loop to fill array list iterating on array
- Refactor the other enhanced for loop to iterate on the array list

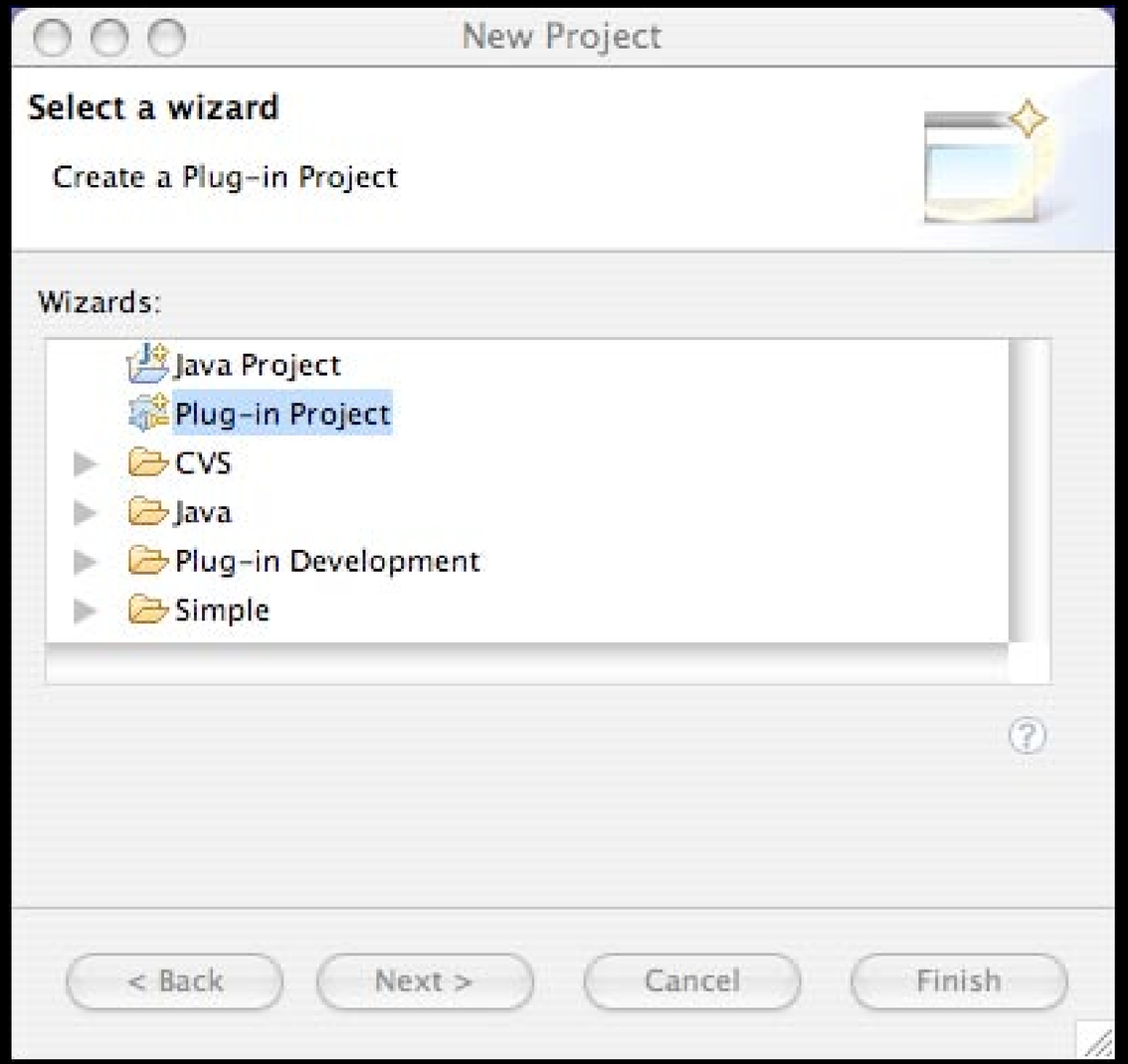
Building Plug-ins

http://developer.apple.com/wwdc2005/

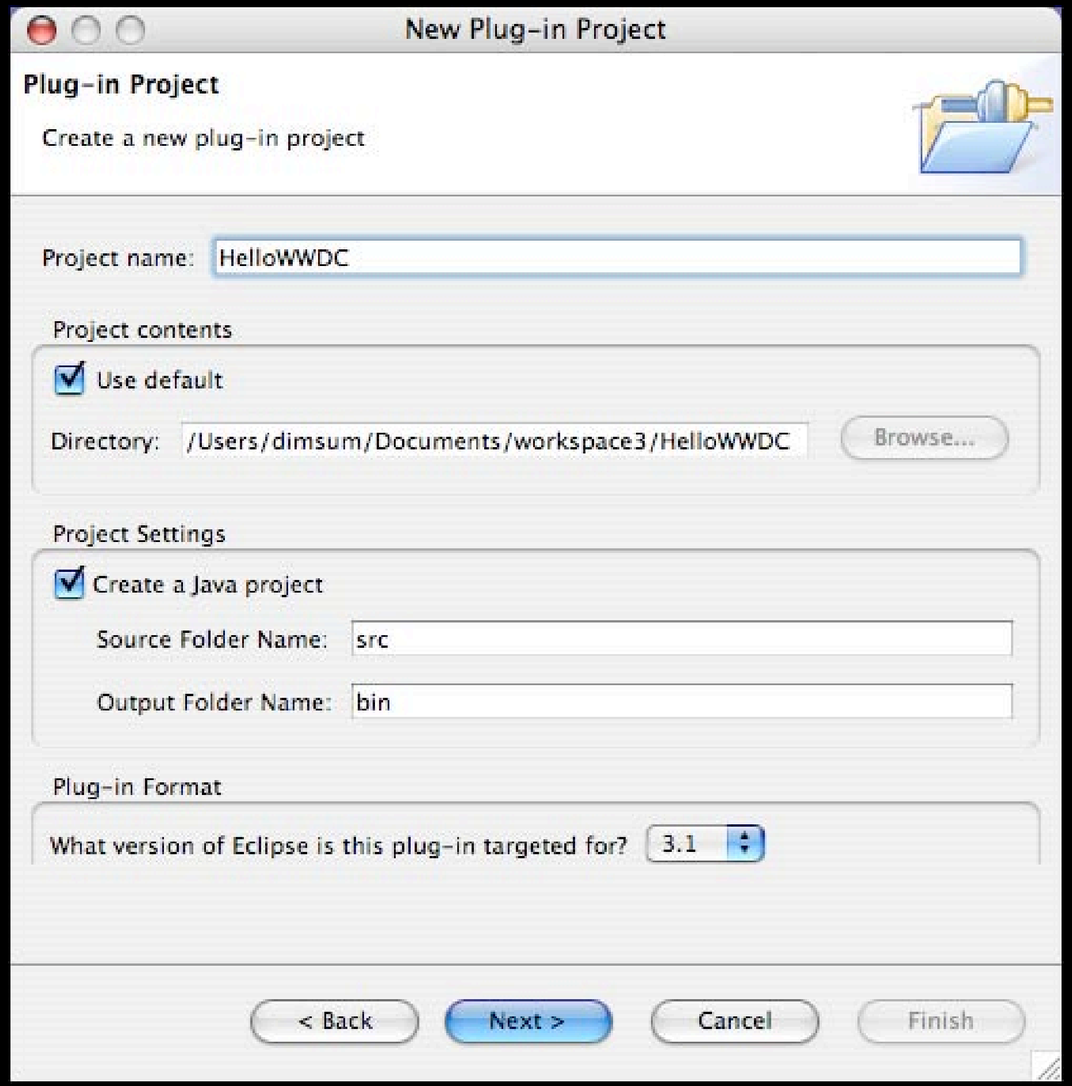
Plug-ins Everywhere



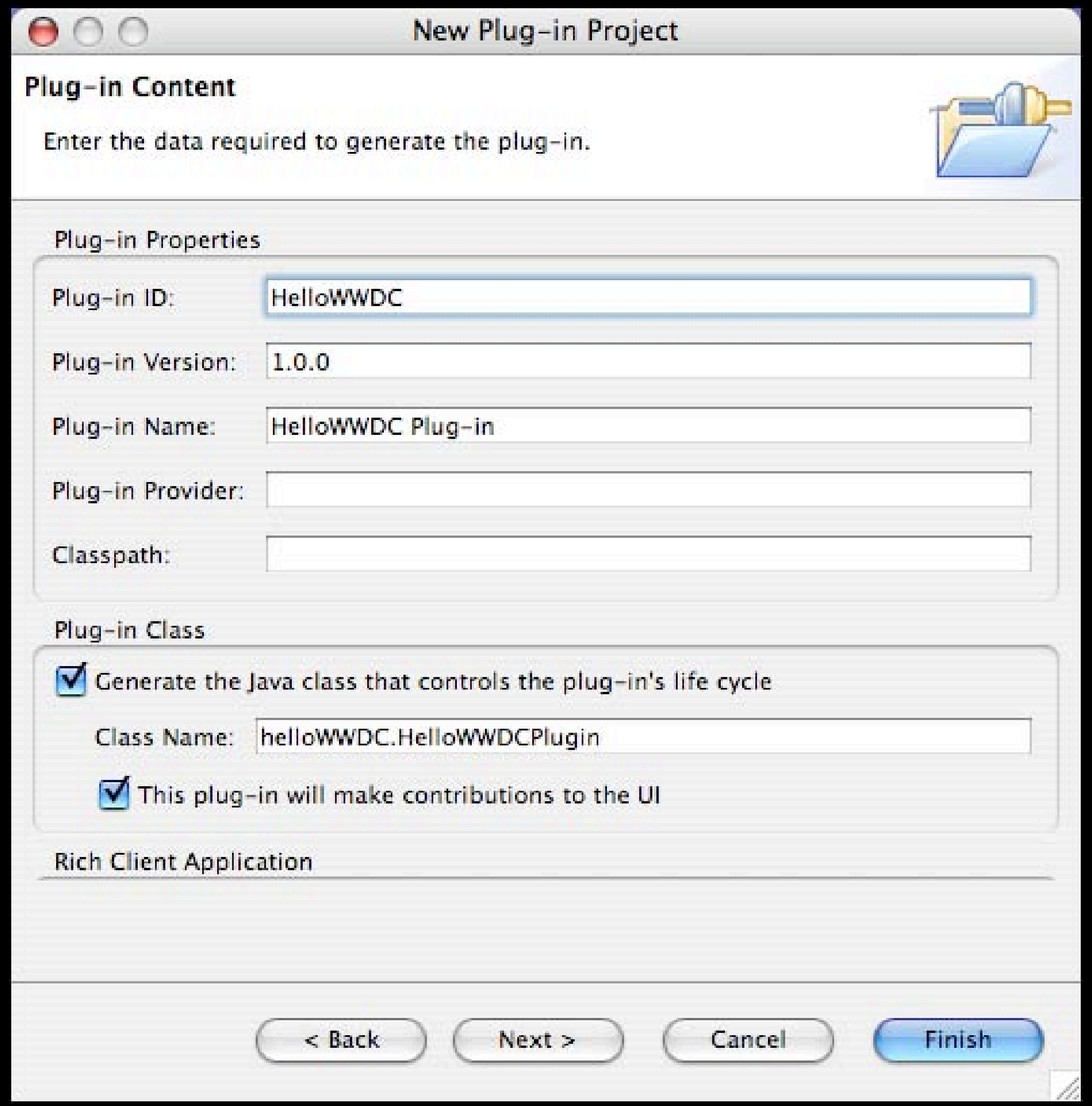
Creating a Plug-in



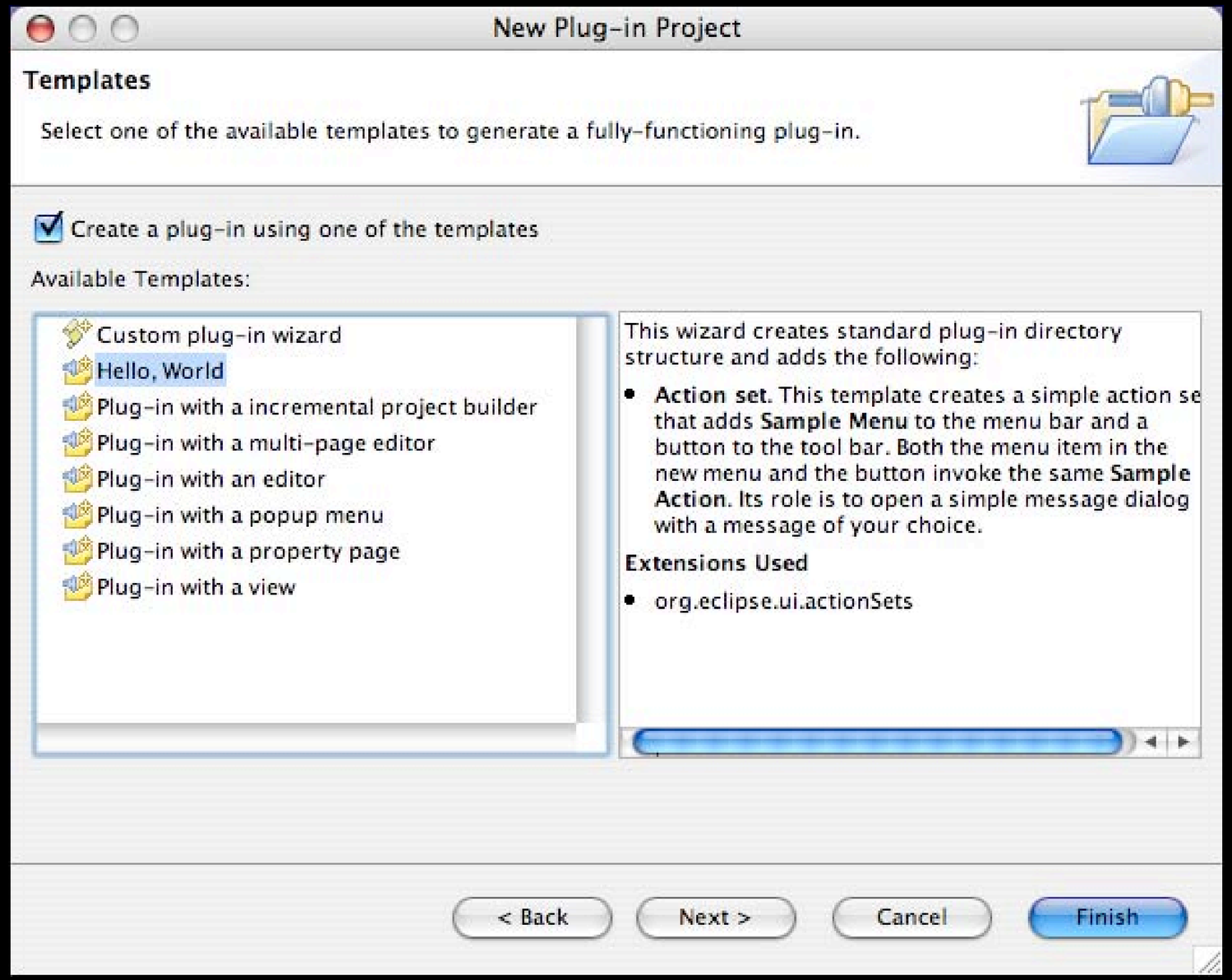
Configuring the Plug-in Project



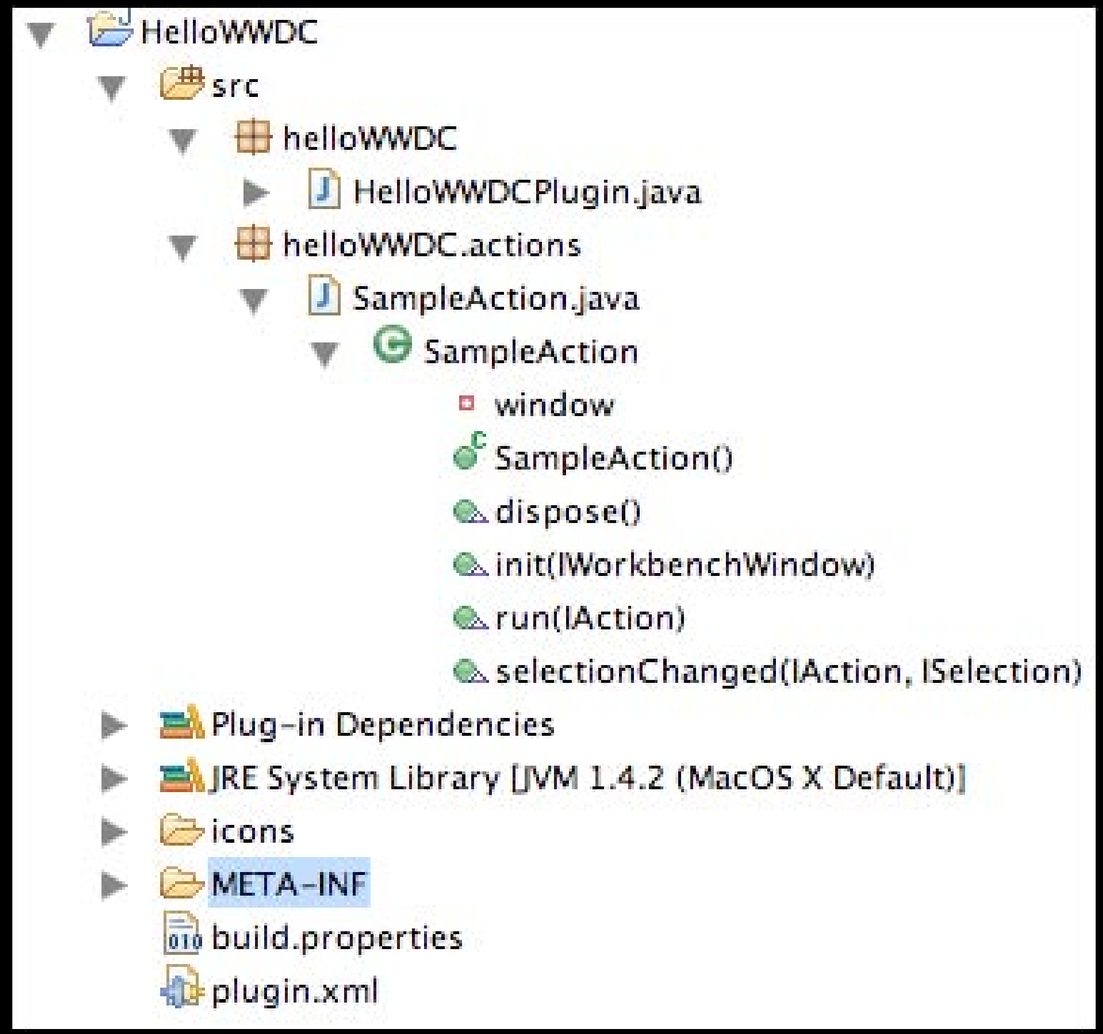
Personalizing the Plug-in



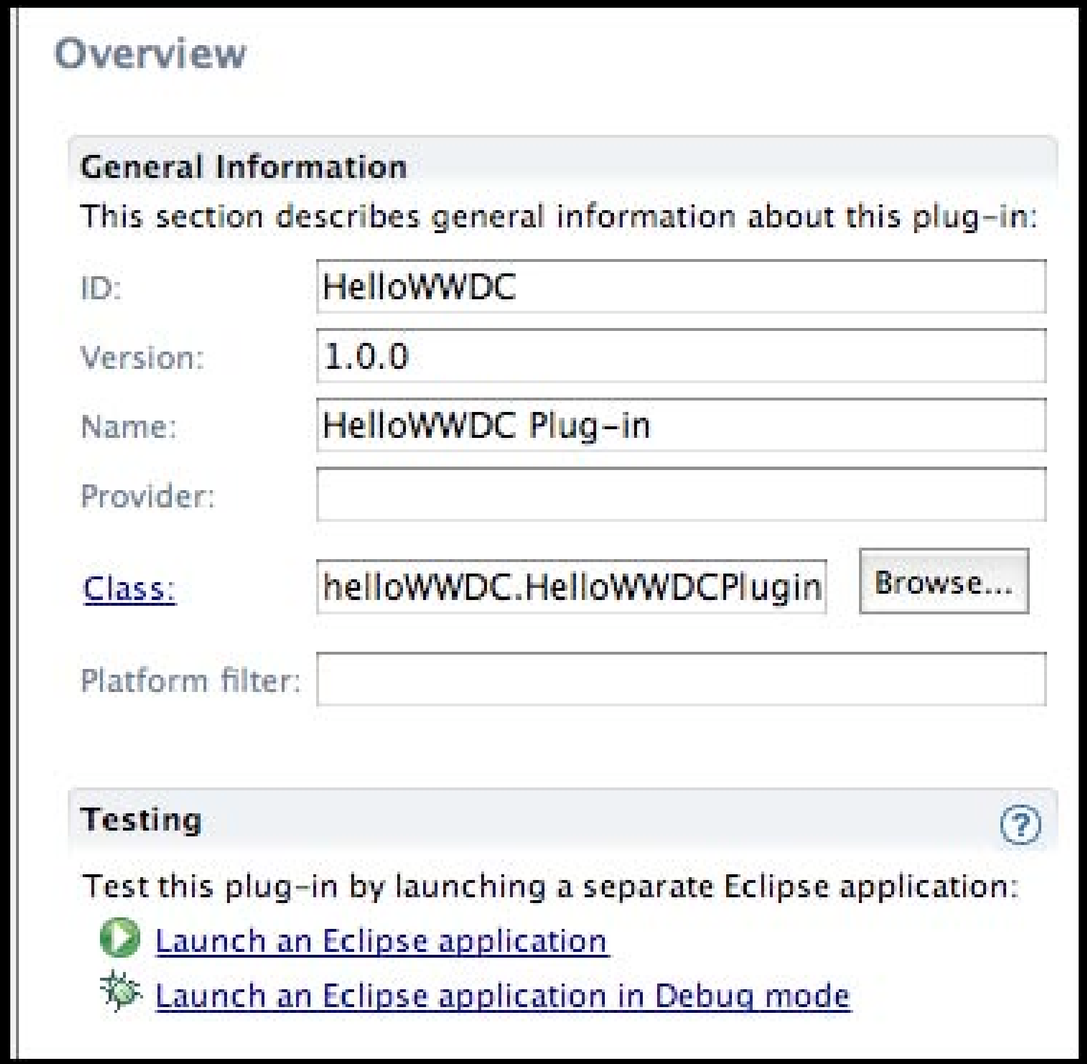
Building on Existing Work



What's Been Generated



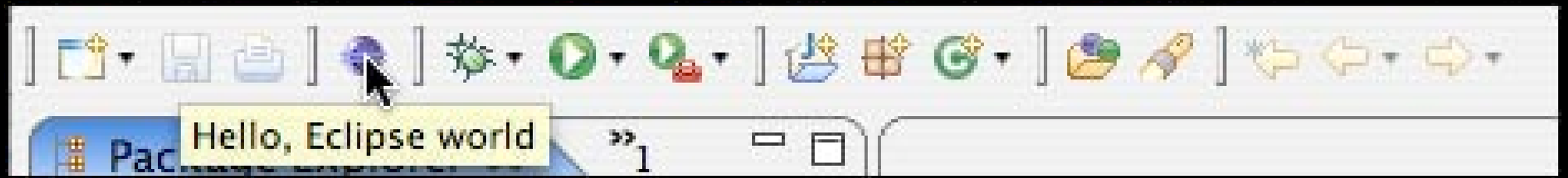
Running the Plug-in



This brings up a second instance of eclipse

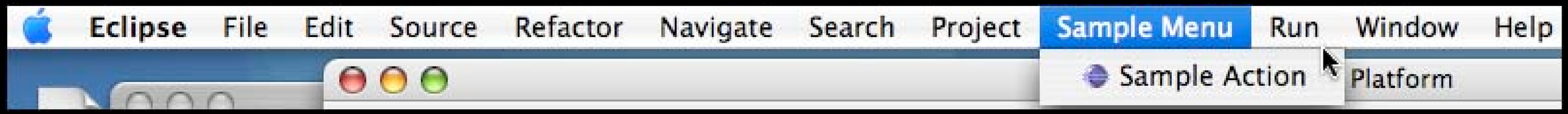
Before and After

Tool Bar

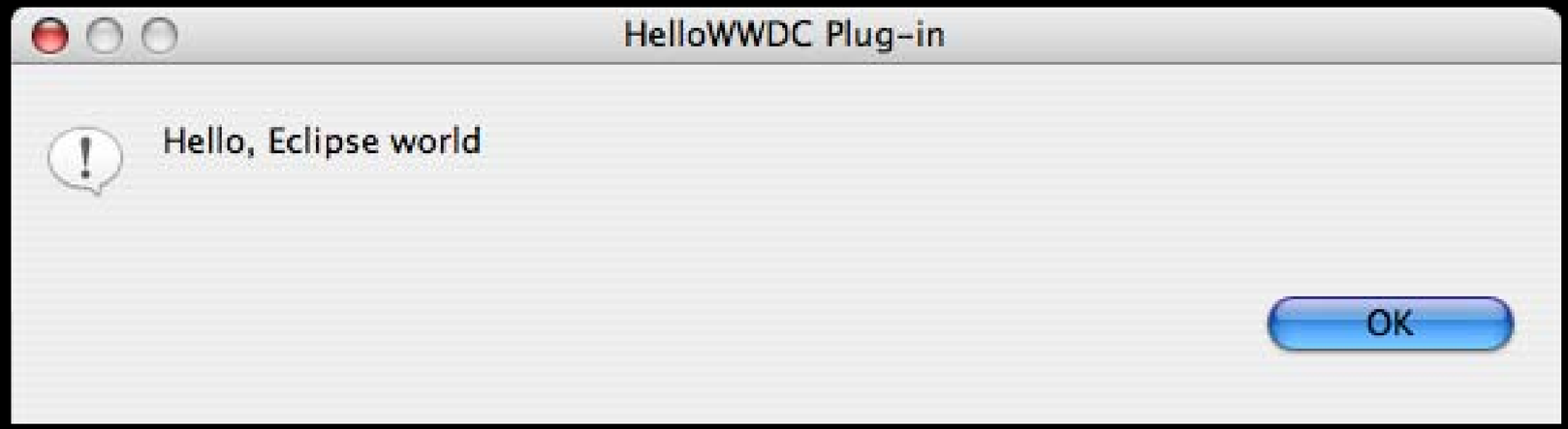


Before and After

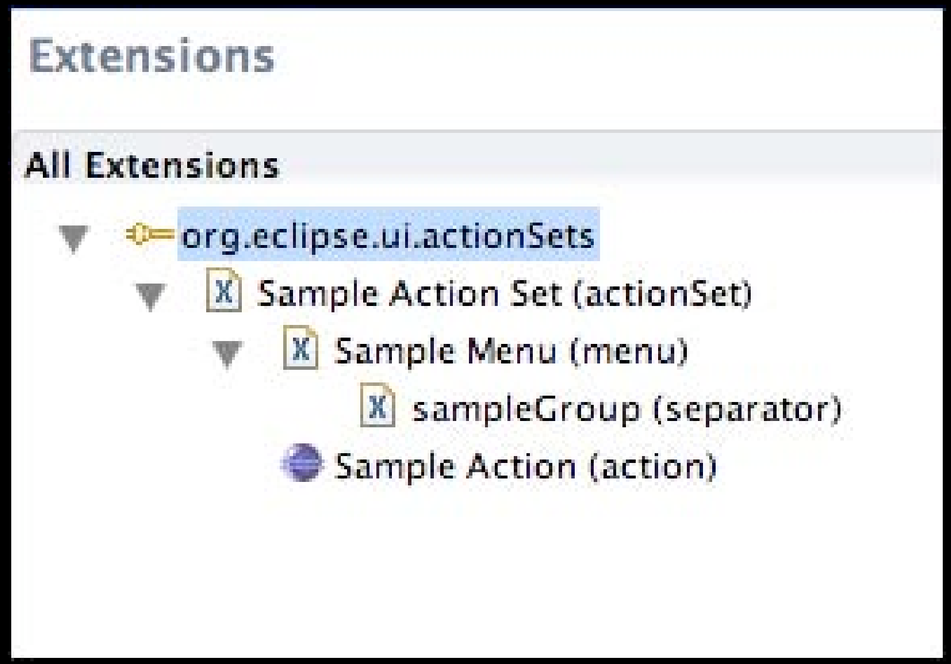
Menu Bar



Run the Plug-in



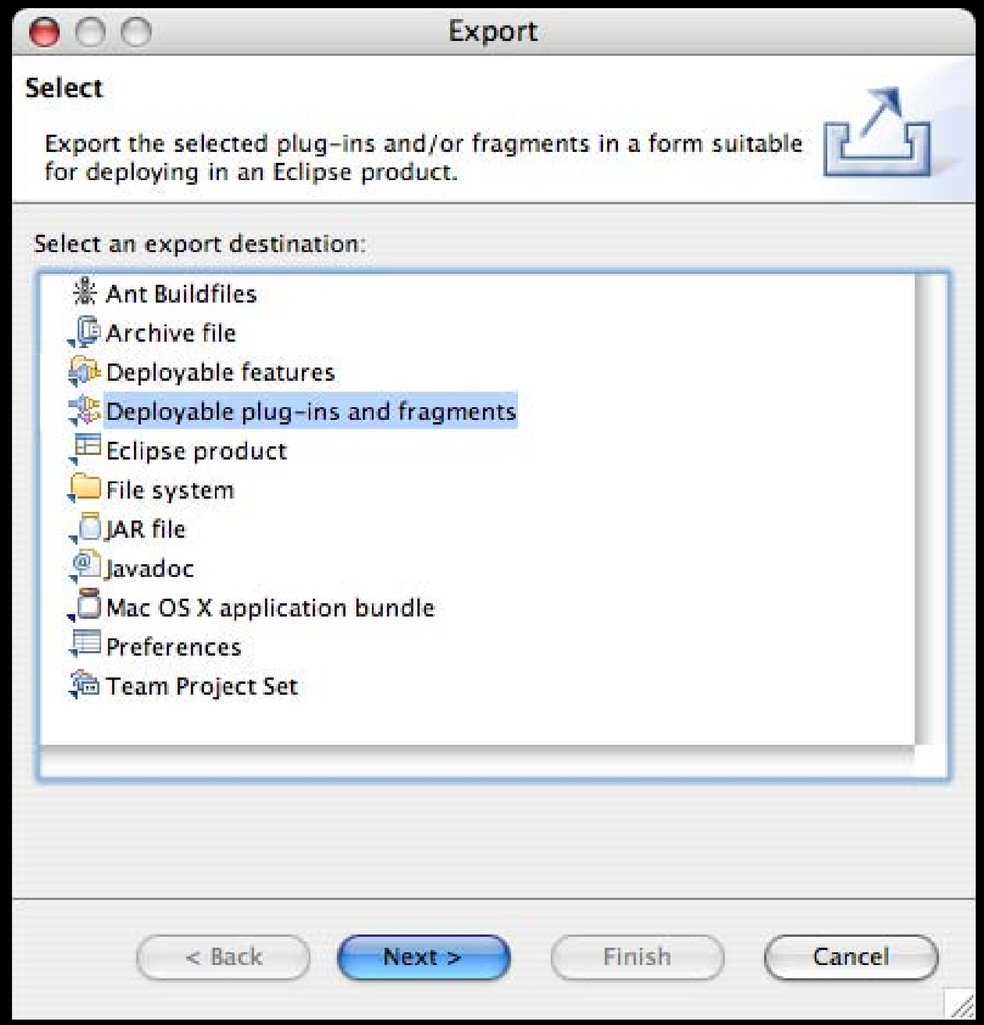
plugin.xml Tree View



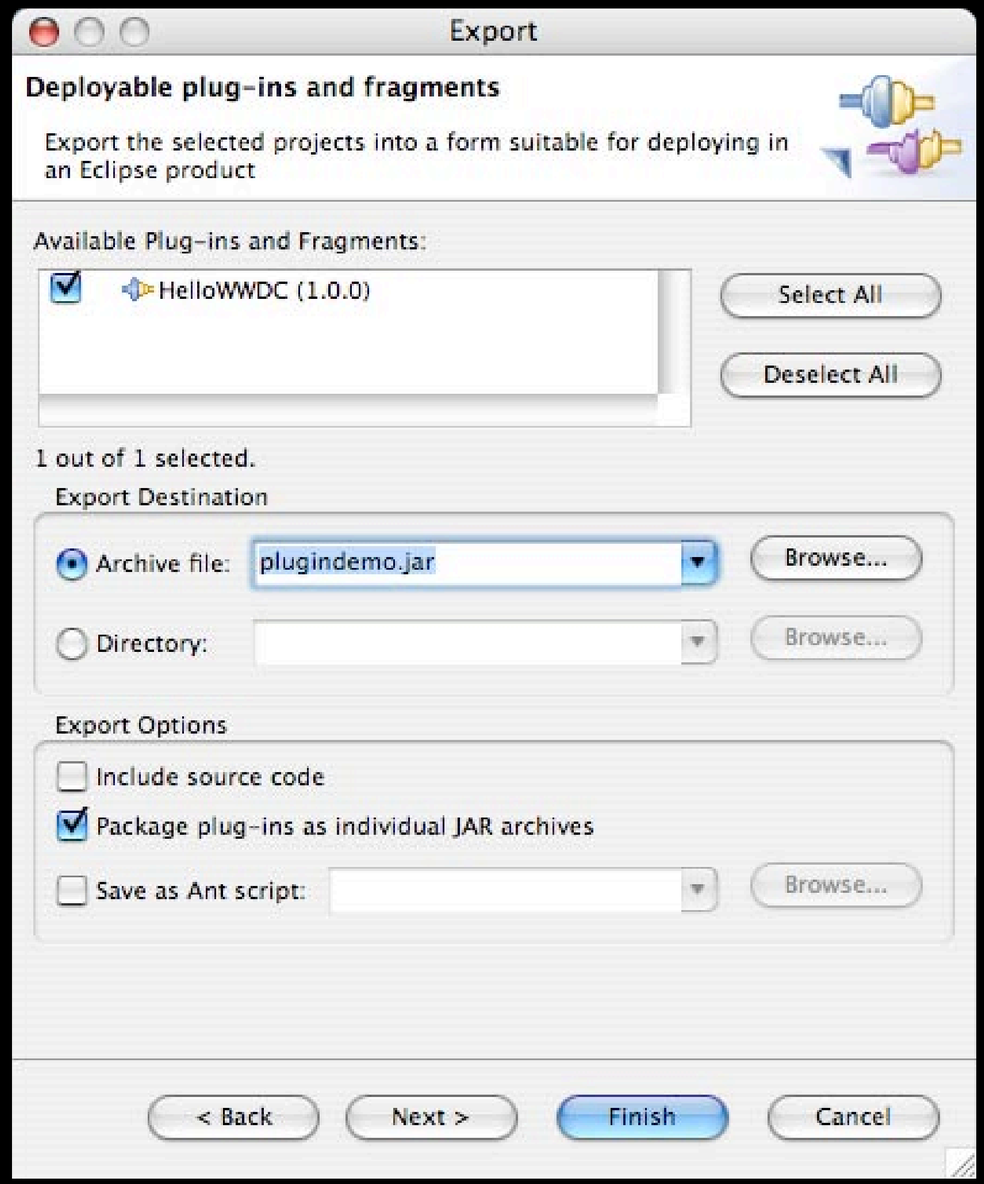
plugin.xml Text View

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin>

   <extension
         point="org.eclipse.ui.actionSets">
      <actionSet
            label="Sample Action Set"
            visible="true"
            id="HelloWWDC.actionSet">
         <menu
               label="Sample &amp;amp;Menu"
               id="sampleMenu">
            <separator
                  name="sampleGroup">
            </separator>
         </menu>
         <action
               label="&amp;amp;Sample Action"
               icon="icons/sample.gif"
               class="helloWWDC.actions.SampleAction"
               tooltip="Hello, Eclipse world"
               menubarPath="sampleMenu/sampleGroup"
               toolbarPath="sampleGroup"
               id="helloWWDC.actions.SampleAction">
         </action>
      </actionSet>
   </extension>

</plugin>
```

Deploying the Plug-in



Deploying the Plug-in



Plug-in Hands-on Demo

- Create your own plug-in project
- Opt for a toolbar element but not a menubar menu
- Change the text in the pop-up window
- Run your plug-in
- Deploy it as a jar file: myplugin.jar
- Use jar xvf myplugin.jar from the command-line to examine the contents of the jar

What We've Done

- Getting started
- Packages and classes
- Programming by intention
- Code assist
- Running tests
- J2SE 5.0
- Building plug-ins

Q&A

Todd Brackett
Global Alliance Manager
tbrackett@apple.com

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference